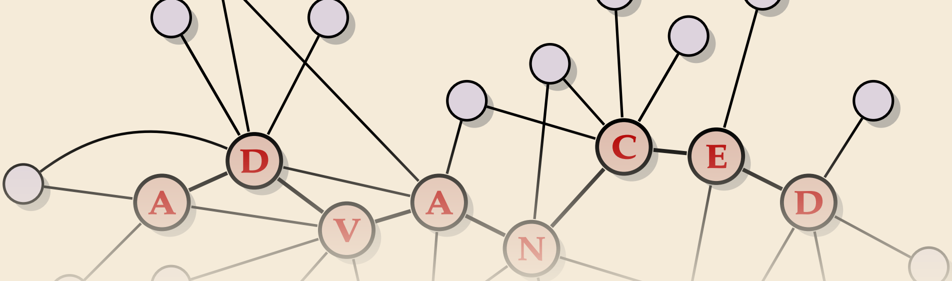



# 6 Advanced Parameterized Ideas

*3 June 2025*

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
*version* `2025-06-03 10:02`

# Outline

## 6 Advanced Parameterized Ideas


6.1 Linear Programs – A Mighty Blackbox Tool
6.2 Linear Programs – Reformulation Tricks
6.3 Linear Programs – The Simplex Algorithm
6.4 Integer Linear Programs
6.5 LP-Based Kernelization
6.6 Lower Bounds by ETH

## 6.1 Linear Programs – A Mighty Blackbox Tool

# Linear Programs

- ***Linear programs (LPs)*** are a class of optimization problems of **continuous** (numerical) variables
- can be exactly solved in worst case polytime ($\textsc{LinearProgramming} \in \mathsf{P}$)
  - interior-point methods, Ellipsoid method
- routinely solved in practice to optimality with millions of variables and constraints
  - Simplex algorithm, interior-point methods
  - many existing solvers, commercial and open source (e. g., HiGHS)

# Hessy James's Apple Farm

- Hessy tries to maximize the profit of his apple farm
  - He is committed to promote regional Hessian heirloom varieties, so he only grows *"Sossenheimer Roter"* and *"Korbacher Edelrenette"*
  - each tree of *"Sossenheimer Roter"* yields apples worth € 195 per year
  - each tree of *"Korbacher Edelrenette"* yields applies worth € 255 per year
  - He has an orchard of 5 000 $m^2$
  - each tree needs 4 $m^2$ of orchard space
  - each tree of *"Sossenheimer Roter"* needs 6 kg of organic fertilizer and 1 h harvest effort per year
  - each tree of *"Korbacher Edelrenette"* needs 4.5 kg of organic fertilizer and 3 h harvest effort per year
  - Hessy can only afford 3000 kg of fertilizer and 1700 h of harvester time per year

# Hessy James's Apple Farm

- Hessy tries to maximize the profit of his apple farm
  - He is committed to promote regional Hessian heirloom varieties, so he only grows *"Sossenheimer Roter"* and *"Korbacher Edelrenette"*
  - each tree of *"Sossenheimer Roter"* yields apples worth € 195 per year
  - each tree of *"Korbacher Edelrenette"* yields applies worth € 255 per year
  - He has an orchard of 5 000 $m^2$
  - each tree needs 4 $m^2$ of orchard space
  - each tree of *"Sossenheimer Roter"* needs 6 kg of organic fertilizer and 1 h harvest effort per year
  - each tree of *"Korbacher Edelrenette"* needs 4.5 kg of organic fertilizer and 3 h harvest effort per year
  - Hessy can only afford 3000 kg of fertilizer and 1700 h of harvester time per year

⇝ How many trees of each variety should Hessy plant?
  - What will constrain us most? Space? Fertilizer? Harvest hours?
  - What profit can Hessy expect?

# Formal Linear Program for Hessy James's Apple Farm

- Classic application of linear programming in *operations research (OR)*
- We formally write LPs as follows:

optimization goal

objective function

constraint

name of the LP

$$
\begin{array}{lrcll}
\textbf{Maximize:} & 195s + 255k & & & \\
\textbf{Subject to:} & 4s + 4k & \leq & 5000 & \text{(Orchard constraint)} \\
 & 6s + 4.5k & \leq & 3000 & \text{(Fertilizer constraint)} \\
 & 1s + 3k & \leq & 1700 & \text{(Harvest constraint)} \\
 & s \phantom{+ 3k} & \geq & 0 & \text{(Non-negativity)} \\
 & k & \geq & 0 & \text{(Non-negativity)}
\end{array}
\qquad (P)
$$

# Formal Linear Program for Hessy James's Apple Farm

- Classic application of linear programming in *operations research (OR)*
- We formally write LPs as follows:

optimization goal

objective function

constraint

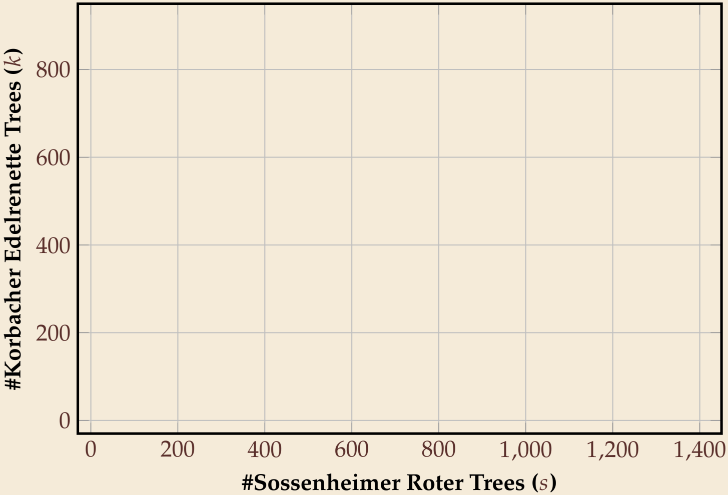
name of the LP

$$\begin{array}{llcll} \textbf{Maximize:} & 195s + 255k & & & \\ \textbf{Subject to:} & 4s + 4k & \le & 5000 & \text{(Orchard constraint)} \\ & 6s + 4.5k & \le & 3000 & \text{(Fertilizer constraint)} \\ & 1s + 3k & \le & 1700 & \text{(Harvest constraint)} \\ & s & \ge & 0 & \text{(Non-negativity)} \\ & k & \ge & 0 & \text{(Non-negativity)} \end{array} \qquad (P)$$

- **Terminology:**
  - $s$ and $k$ are the two *variables* of the problem; these are always real numbers.
  - A vector $(s, k) \in \mathbb{R}^2$ is a *feasible solution* for the LP if it satisfied all constraints.
  - The largest value of the objective function (over all feasible solutions) is the *(optimal) value* $z^*$ of the LP
  - A feasible solution $(s^*, k^*) \in R^2$ with optimal objective value $z^*$ is called an *optimal solution*

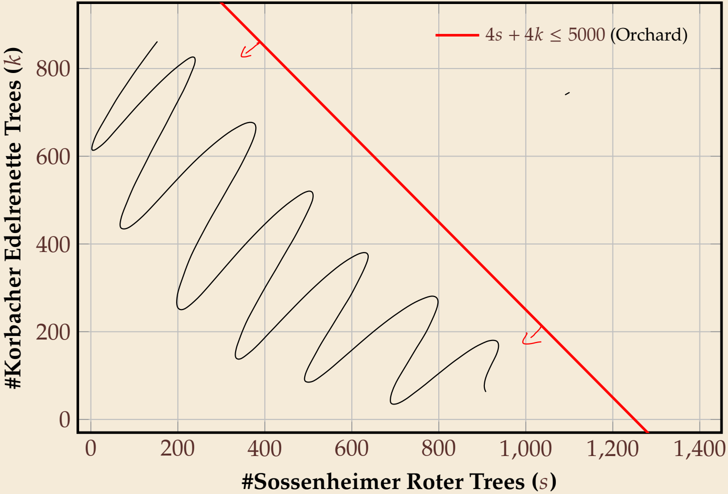
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



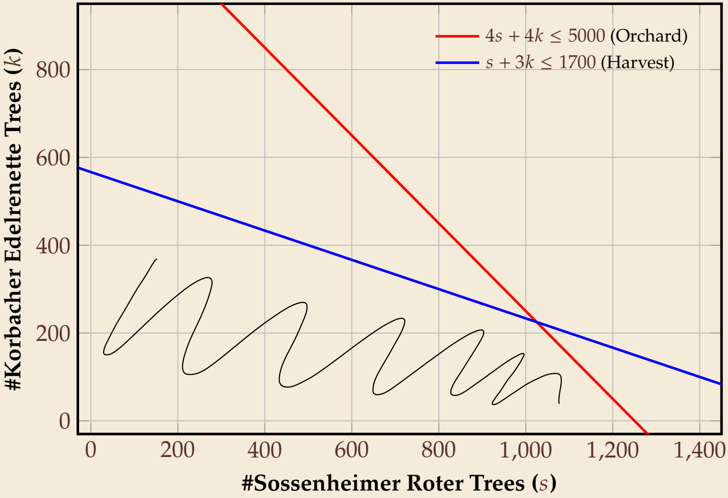
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



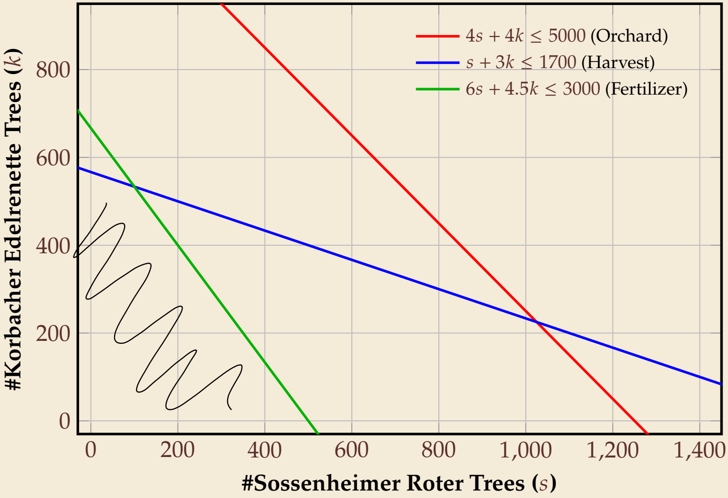
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



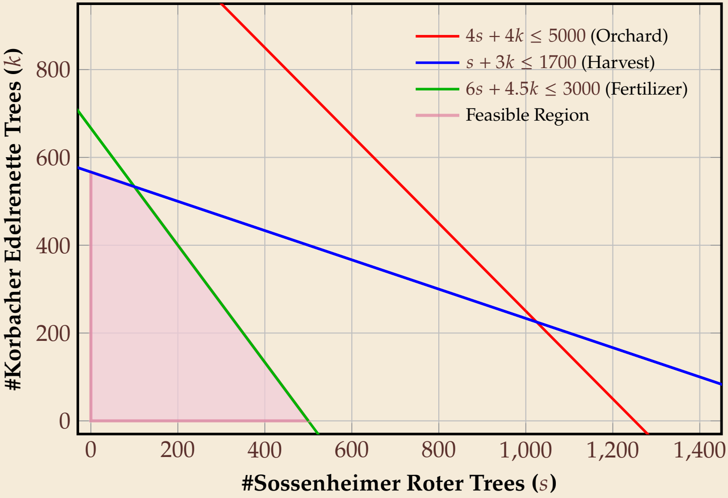
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



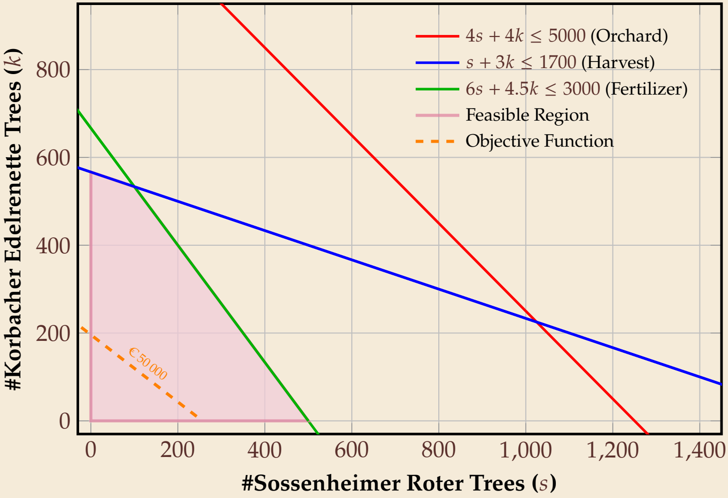
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



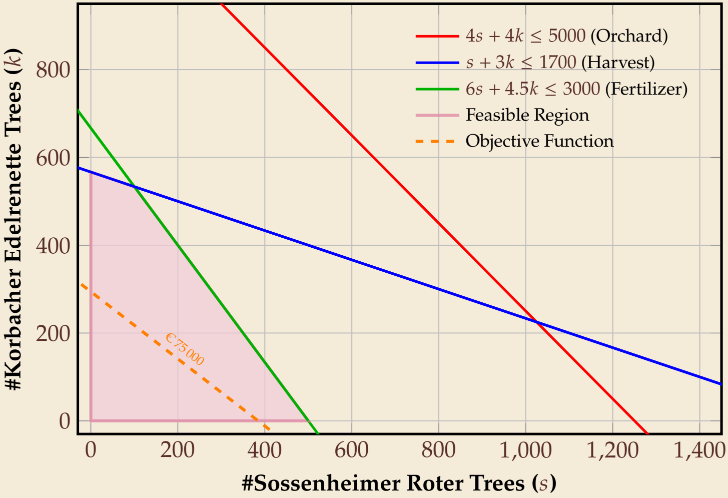
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



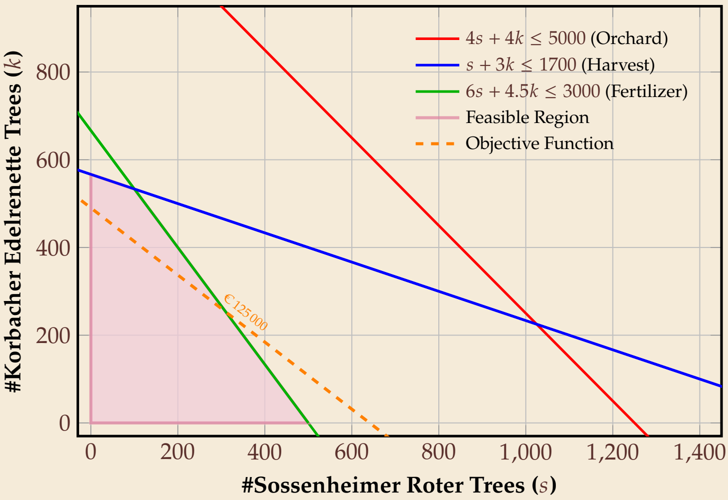
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



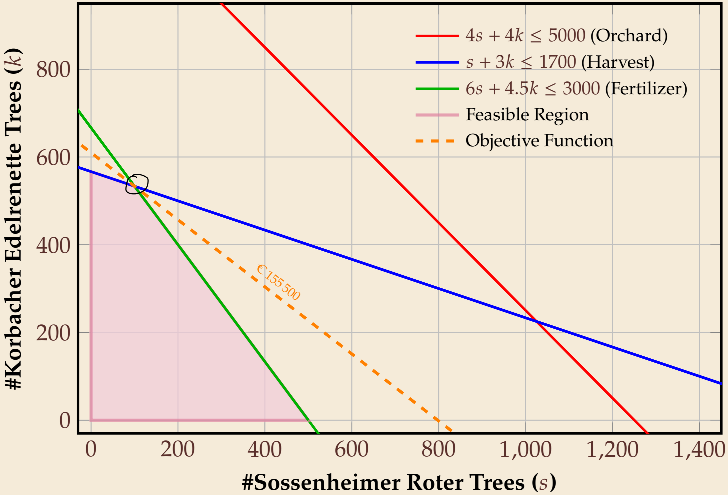
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



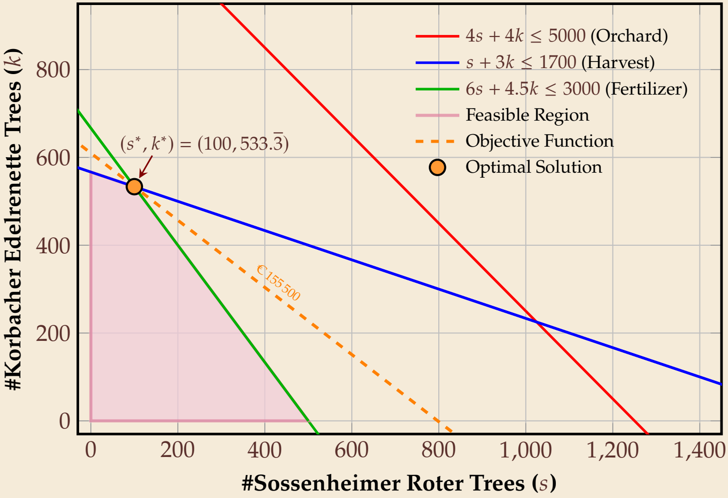
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



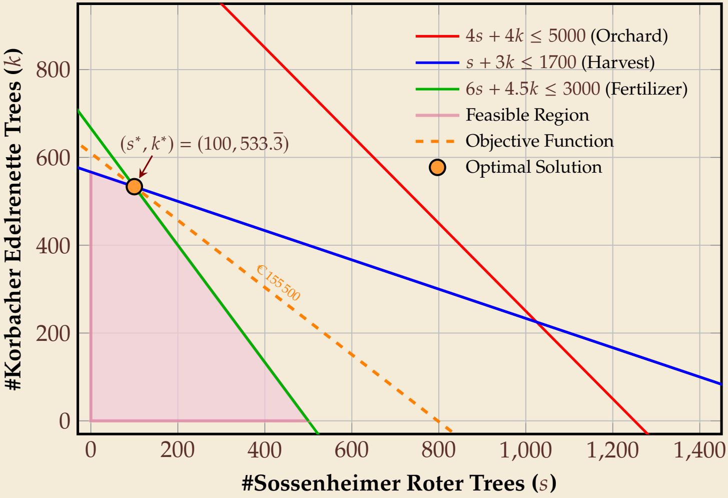
# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically



# 2D LPs – Graphical Solution

LPs with **two** variables can be solved graphically




⇝ Hessy should plant

- 100 *Sossenheimer Roter* trees and
- hmm . . .
- $533+\frac{1}{3}$ *Korbacher Edelrenette* trees
- Harvest **and** fertilizer *tight*
- orchard space isn't

⇝ know what to change

# LPs – The General Case

- General LP:

$$\begin{array}{rrcll}
\min & c_1x_1 + \cdots + \;\; c_nx_n & & & \\
\text{s. t.} & a_{i,1}x_1 + \cdots + a_{i,n}x_n & = & b_i & (\text{for } i = 1, \ldots, p) \\
& a_{i,1}x_1 + \cdots + a_{i,n}x_n & \leq & b_i & (\text{for } i = p+1, \ldots, q) \\
& a_{i,1}x_1 + \cdots + a_{i,n}x_n & \geq & b_i & (\text{for } i = q+1, \ldots, m) \\
& x_j & \geq & 0 & (\text{for } j = 1 \ldots, r) \\
& x_j & \lesseqgtr & 0 & (\text{for } j = r+1 \ldots, n)
\end{array}$$

"don't care" (just to make it explicit)

  - arbitrary **linear** objective function
  - arbitrary **linear** constraints, of type "=", "≤" or "≥"
  - variables with non-negativity constraint and unconstrained variables

# LPs – The General Case

- General LP:

$$\begin{array}{rrcll} \min & c_1 x_1 + \cdots + \phantom{a_{i,}} c_n x_n & & & \\ \text{s.t.} & a_{i,1}x_1 + \cdots + a_{i,n}x_n & = & b_i & (\text{for } i = 1, \ldots, p) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \le & b_i & (\text{for } i = p+1, \ldots, q) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \ge & b_i & (\text{for } i = q+1, \ldots, m) \\ & x_j & \ge & 0 & (\text{for } j = 1 \ldots, r) \\ & x_j & \lesseqgtr & 0 & (\text{for } j = r+1 \ldots, n) \end{array}$$

"don't care" (just to make it explicit)

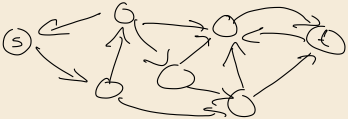
  - arbitrary **linear** objective function
  - arbitrary **linear** constraints, of type "=", "≤" or "≥"
  - variables with non-negativity constraint and unconstrained variables

- In general, an LP can
  (a) have a *finite* optimal *objective value*
  (b) be *infeasible* (contradictory constraints / empty feasibility region), or
  (c) be *unbounded* (allow arbitrarily small objective values "$-\infty$")

⇝ in polytime, can detect which case applies **and** compute optimal solution in case (a)

# Classic Modeling Example – Max Flow

- The maximum-$s$-$t$-flow problem in a graph $G = (V, E)$ can be reduced to an LP (Flow)
  - variable $f_e$ for each edge $e \in E$
  - maximize flow value $F$ = flow out of $s$
  - constraint for edge capacity $C(e)$ at each edge
  - constraint for flow conservation at each vertex $v$ (except $s$ and $t$)




$v \to w \quad (v,w)$

$$
\begin{aligned}
\max \quad & F \\
\text{s. t.} \quad & F = \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} \\
& f_{vw} \le C(vw) && (\text{for } vw \in E) \\
& \sum_{w \in V} f_{wv} = \sum_{w \in V} f_{vw} && (\text{for } v \in V \setminus \{s, t\}) \\
& f_e \ge 0 && (\text{for } e \in E)
\end{aligned}
\qquad (\text{Flow})
$$

## 6.2 Linear Programs – Reformulation Tricks

# How to solve an LP?

- Our focus will be on using LPs as a tool
  - in theory: reducing problem to an LP means polytime solvable
  - in practice: call good solver!

# How to solve an LP?

- Our focus will be on using LPs as a tool
  - in theory: reducing problem to an LP means polytime solvable
  - in practice: call good solver!

- *But as with any good tool, it helps to gave an idea of* ***how*** *it works to effectively use it*

$\rightsquigarrow$ We will briefly visit the conceptual ideas of the simplex algorithm

# Recall: General Form of LPs

- General LP:

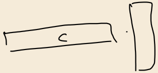
$$
\begin{array}{rrcll}
\min & c_1 x_1 + \cdots + \; c_n x_n & & & \\
\text{s.t.} & a_{i,1} x_1 + \cdots + a_{i,n} x_n & = & b_i & (\text{for } i = 1, \ldots, p) \\
& a_{i,1} x_1 + \cdots + a_{i,n} x_n & \leq & b_i & (\text{for } i = p+1, \ldots, q) \\
& a_{i,1} x_1 + \cdots + a_{i,n} x_n & \geq & b_i & (\text{for } i = q+1, \ldots, m) \\
& x_j & \geq & 0 & (\text{for } j = 1 \ldots, r) \\
& x_j & \lesseqgtr & 0 & (\text{for } j = r+1 \ldots, n)
\end{array}
$$

  - linear objective function and constraints ("$=$", "$\leq$", or "$\geq$")
  - variables with non-negativity constraint and unconstrained variables

- **Conventions:**
  - $n$ variables (always called $x_j$)
  - $m$ constraints (coefficients always called $a_{i,j}$, right-hand sides $b_i$)
  - minimize objective ("cost" ), coefficients $c_j$;   objective value $z = c_1 x_1 + \cdots c_n x_n$

# Enter Linear Algebra

$$
\begin{array}{llcll}
\min & c_1x_1 + \cdots + \ c_nx_n & & & \\
\text{s. t.} & a_{i,1}x_1 + \cdots + a_{i,n}x_n & = & b_i & (\text{for } i = 1, \ldots, p) \\
& a_{i,1}x_1 + \cdots + a_{i,n}x_n & \le & b_i & (\text{for } i = p+1, \ldots, q) \\
& a_{i,1}x_1 + \cdots + a_{i,n}x_n & \ge & b_i & (\text{for } i = q+1, \ldots, m) \\
& x_j & \ge & 0 & (\text{for } j = 1 \ldots, r) \\
& x_j & \lesseqgtr & 0 & (\text{for } j = r+1 \ldots, n)
\end{array}
$$




- Spelling out all those linear combinations is cumbersome

$\rightsquigarrow$ Concise notation via **matrix and vector products**

- We write
  - **variables** $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ **cost** coefficients $\boldsymbol{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n$ (bold $\rightsquigarrow$ vector/matrix) $\rightsquigarrow$ **objective**: $\min \boldsymbol{c}^T \cdot \boldsymbol{x}$

# Enter Linear Algebra

$$
\begin{array}{llcll}
\min & c_1x_1 + \cdots + \ c_nx_n & & & \\
\text{s.t.} & a_{i,1}x_1 + \cdots + a_{i,n}x_n & = & b_i & (\text{for } i = 1, \ldots, p) \\
 & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \leq & b_i & (\text{for } i = p+1, \ldots, q) \\
 & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \geq & b_i & (\text{for } i = q+1, \ldots, m) \\
 & x_j & \geq & 0 & (\text{for } j = 1 \ldots, r) \\
 & x_j & \lesseqgtr & 0 & (\text{for } j = r+1 \ldots, n)
\end{array}
$$

- Spelling out all those linear combinations is cumbersome

$\rightsquigarrow$ Concise notation via **matrix and vector products**

- We write
  - **variables** $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ **cost** coefficients $\underbrace{\boldsymbol{c}}_{}$ $= \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n$ $\rightsquigarrow$ **objective**: $\min \boldsymbol{c}^T \cdot \boldsymbol{x}$

bold $\rightsquigarrow$ vector/matrix

transpose

dot product / scalar product

# Enter Linear Algebra

$$\begin{array}{llcll} \min & c_1x_1 + \cdots + c_nx_n & & & \\ \text{s.t.} & a_{i,1}x_1 + \cdots + a_{i,n}x_n & = & b_i & (\text{for } i = 1,\ldots,p) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \leq & b_i & (\text{for } i = p+1,\ldots,q) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \geq & b_i & (\text{for } i = q+1,\ldots,m) \\ & x_j & \geq & 0 & (\text{for } j = 1\ldots,r) \\ & x_j & \lessgtr & 0 & (\text{for } j = r+1\ldots,n) \end{array}$$

- Spelling out all those linear combinations is cumbersome

$\rightsquigarrow$ Concise notation via **matrix and vector products**

- We write
  - **variables** $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ bold $\rightsquigarrow$ vector/matrix **cost** coefficients $\boldsymbol{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n$ $\rightsquigarrow$ **objective**: $\min \boldsymbol{c}^T \cdot \boldsymbol{x}$
    transpose
    dot product / scalar product
  - **"="-constraints**

$$A^{(=)} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p,1} & a_{p,2} & \cdots & a_{p,n} \end{pmatrix} \in \mathbb{R}^{p\times n} \qquad \boldsymbol{b}^{(=)} = \begin{pmatrix} b_1 \\ \vdots \\ b_p \end{pmatrix} \in \mathbb{R}^p \qquad \rightsquigarrow \quad A^{(=)} \cdot \boldsymbol{x} = \boldsymbol{b}^{(=)}$$

# Enter Linear Algebra

$$\begin{array}{lrcll} \min & c_1x_1 + \cdots + \ c_nx_n & & & \\ \text{s.t.} & a_{i,1}x_1 + \cdots + a_{i,n}x_n & = & b_i & (\text{for } i = 1,\ldots,p) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \le & b_i & (\text{for } i = p+1,\ldots,q) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \ge & b_i & (\text{for } i = q+1,\ldots,m) \\ & x_j & \ge & 0 & (\text{for } j = 1\ldots,r) \\ & x_j & \lesseqgtr & 0 & (\text{for } j = r+1\ldots,n) \end{array}$$

- Spelling out all those linear combinations is cumbersome

⇝ Concise notation via **matrix and vector products**

- We write
  - **variables** $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ **cost** coefficients $\boldsymbol{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n$ ⇝ **objective:** $\min \boldsymbol{c}^T \cdot \boldsymbol{x}$

    bold ⇝ vector/matrix

    transpose

    dot product / scalar product
  - **"="-constraints**

    $$\boldsymbol{A}^{(=)} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p,1} & a_{p,2} & \cdots & a_{p,n} \end{pmatrix} \in \mathbb{R}^{p\times n} \qquad \boldsymbol{b}^{(=)} = \begin{pmatrix} b_1 \\ \vdots \\ b_p \end{pmatrix} \in \mathbb{R}^p \quad \rightsquigarrow \quad \boldsymbol{A}^{(=)} \cdot \boldsymbol{x} \ = \ \boldsymbol{b}^{(=)}$$
  - similarly for "≤" and "≥" constraints: $\boldsymbol{A}^{(\le)}\boldsymbol{x} \le \boldsymbol{b}^{(\le)}$ and $\boldsymbol{A}^{(\ge)}\boldsymbol{x} \ge \boldsymbol{b}^{(\ge)}$

    elementwise ≤

# Enter Linear Algebra

$$\begin{array}{llcll} \min & c_1x_1 + \cdots + c_nx_n & & & \\ \text{s.t.} & a_{i,1}x_1 + \cdots + a_{i,n}x_n & = & b_i & (\text{for } i = 1,\dots,p) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \le & b_i & (\text{for } i = p+1,\dots,q) \\ & a_{i,1}x_1 + \cdots + a_{i,n}x_n & \ge & b_i & (\text{for } i = q+1,\dots,m) \\ & x_j & \ge & 0 & (\text{for } j = 1\dots,r) \\ & x_j & \lesseqgtr & 0 & (\text{for } j = r+1\dots,n) \end{array}$$

- Spelling out all those linear combinations is cumbersome

⇝ Concise notation via **matrix and vector products**

- We write
  - **variables** $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$    **cost** coefficients $\boldsymbol{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n$    ⇝ **objective**: $\min \boldsymbol{c}^T \cdot \boldsymbol{x}$

    bold ⇝ vector/matrix

    transpose

    dot product / scalar product
  - **"="-constraints**

    $$\boldsymbol{A}^{(=)} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p,1} & a_{p,2} & \cdots & a_{p,n} \end{pmatrix} \in \mathbb{R}^{p\times n} \qquad \boldsymbol{b}^{(=)} = \begin{pmatrix} b_1 \\ \vdots \\ b_p \end{pmatrix} \in \mathbb{R}^p \qquad \rightsquigarrow \quad \boldsymbol{A}^{(=)} \cdot \boldsymbol{x} = \boldsymbol{b}^{(=)}$$
  - similarly for "≤" and "≥" constraints: $\boldsymbol{A}^{(\le)}\boldsymbol{x} \le \boldsymbol{b}^{(\le)}$ and $\boldsymbol{A}^{(\ge)}\boldsymbol{x} \ge \boldsymbol{b}^{(\ge)}$

    elementwise ≤

  ⇝ a **single** constraint $i$ can be written as $\boldsymbol{A}_{i,\bullet}\,\boldsymbol{x} = b_i$    A[i,:]

  (generally write $\boldsymbol{A}_{i,\bullet}$ for the $i$th row of $\boldsymbol{A}$ and $\boldsymbol{A}_{\bullet,j}$ for the $j$th column)

# Reformulations

Tricks of the Trade for working with LPs:

- min suffices: $\max \boldsymbol{c}^T \boldsymbol{x} \;=\; -\min(-\boldsymbol{c})^T \boldsymbol{x}$
- "≥"-constraints: $A_{i,\bullet}\,\boldsymbol{x} \;\geq\; b_i \quad\Longleftrightarrow\quad (-A)_{i,\bullet}\,\boldsymbol{x} \;\leq\; -b_i$

# Reformulations

Tricks of the Trade for working with LPs:

- min suffices: $\max \boldsymbol{c}^T \boldsymbol{x} \;=\; -\min(-\boldsymbol{c})^T \boldsymbol{x}$
- "≥"-constraints: $A_{i,\bullet}\,\boldsymbol{x} \;\geq\; b_i \quad\Longleftrightarrow\quad (-A)_{i,\bullet}\,\boldsymbol{x} \;\leq\; -b_i$
- *slack variables*: $A_{i,\bullet}\,\boldsymbol{x} \;\leq\; b_i \quad\Longleftrightarrow\quad A_{i,\bullet}\,\boldsymbol{x} + \boldsymbol{x_{s_i}} \;=\; b_i \quad \text{and} \quad x_{s_i} \;\geq\; 0$

  ($x_{s_i}$ is a new additional variable)

# Reformulations

Tricks of the Trade for working with LPs:

- min suffices: $\max \boldsymbol{c}^T \boldsymbol{x} \; = \; -\min(-\boldsymbol{c})^T \boldsymbol{x}$
- "≥"-constraints: $A_{i,\bullet}\,\boldsymbol{x} \; \geq \; b_i \quad \Longleftrightarrow \quad (-A)_{i,\bullet}\,\boldsymbol{x} \; \leq \; -b_i$
- *slack variables*: $A_{i,\bullet}\,\boldsymbol{x} \; \leq \; b_i \quad \Longleftrightarrow \quad A_{i,\bullet}\,\boldsymbol{x} + \boldsymbol{x_{s_i}} \; = \; b_i \quad \text{and} \quad x_{s_i} \; \geq \; 0$

  ($x_{s_i}$ is a new additional variable)
- *nonnegative*: variable $x_j \; \lesseqgtr \; 0 \quad \Longleftrightarrow \quad x_j \; = \; x_{j,+} - x_{j,-} \quad \text{and} \quad x_{j,+}, x_{j,-} \; \geq \; 0$

  ($x_{j,+}$ and $x_{j,-}$ are new additional variables)

# Reformulations

Tricks of the Trade for working with LPs:

- min suffices: $\max \boldsymbol{c}^T\boldsymbol{x} = -\min(-\boldsymbol{c})^T\boldsymbol{x}$
- "$\ge$"-constraints: $A_{i,\bullet}\boldsymbol{x} \ge b_i \iff (-A)_{i,\bullet}\boldsymbol{x} \le -b_i$
- *slack variables*: $A_{i,\bullet}\boldsymbol{x} \le b_i \iff A_{i,\bullet}\boldsymbol{x} + \boldsymbol{x}_{s_i} = b_i$ and $x_{s_i} \ge 0$

  ($x_{s_i}$ is a new additional variable)
- *nonnegative*: variable $x_j \lesseqgtr 0 \iff x_j = x_{j,+} - x_{j,-}$ and $x_{j,+}, x_{j,-} \ge 0$

  ($x_{j,+}$ and $x_{j,-}$ are new additional variables)

⇝ To solve LPs, can assume one of the following **normal forms**

$$\boxed{\begin{aligned} \min\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & A\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{aligned}} \quad \text{or} \quad \boxed{\begin{aligned} \min\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{aligned}}$$

with $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{c} \in \mathbb{R}^n$

$m \geq n$ $\qquad$ $m \lessgtr n$

## 6.3 Linear Programs – The Simplex Algorithm

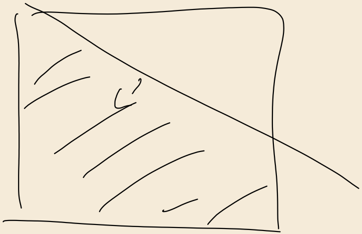
# Simplex – Geometric Intuition

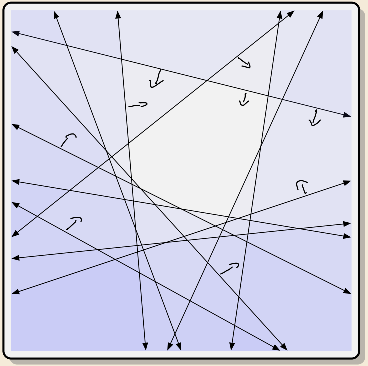
$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax \le b \\ & x \ge 0 \end{aligned}$$
+ *nondegeneracy*

- constraint $A_{i,\bullet}x \le b_i$ defines a *hyperplane*

$\rightsquigarrow$ *halfspace* $H_i = \{x \in \mathbb{R}^n : A_{i,\bullet}x \le b_i\}$

$\boxed{n=2}$





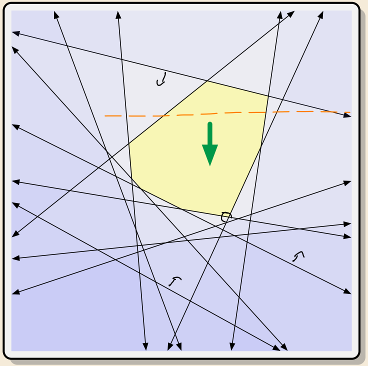
# Simplex – Geometric Intuition

$$\begin{aligned} \min\ & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.}\ & A\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{aligned}$$
+ *nondegeneracy*

- constraint $A_{i,\bullet}\boldsymbol{x} \le b_i$ defines a *hyperplane*

$\rightsquigarrow$ *halfspace* $H_i = \{\boldsymbol{x} \in \mathbb{R}^n : A_{i,\bullet}\boldsymbol{x} \le b_i\}$

- $\boldsymbol{c}$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{c}^T\boldsymbol{x} = 0\}$)
  - *"Roll a ball downhill inside feasible region"*



# Simplex – Geometric Intuition

$$\begin{aligned} \min\ & \mathbf{c}^T\mathbf{x} \\ \text{s.t.}\ & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0} \end{aligned}$$
+ *nondegeneracy*

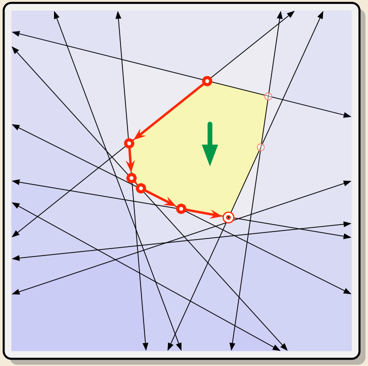
- constraint $A_{i,\bullet}\mathbf{x} \le b_i$ defines a *hyperplane*

$\rightsquigarrow$ *halfspace* $H_i = \{\mathbf{x} \in \mathbb{R}^n : A_{i,\bullet}\mathbf{x} \le b_i\}$

- $\mathbf{c}$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{c}^T\mathbf{x} = 0\}$)
  - *"Roll a ball downhill inside feasible region"*

  $\rightsquigarrow$ Optimal point $\mathbf{x}^*$ must lie on boundary!

  (assuming finite optimal objective value $z^*$)

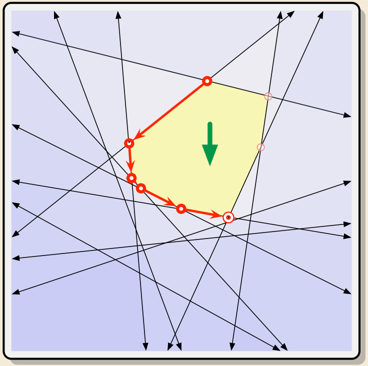


# Simplex – Geometric Intuition

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$
+ *nondegeneracy*

- constraint $A_{i,\bullet}\boldsymbol{x} \leq b_i$ defines a *hyperplane*

$\rightsquigarrow$ *halfspace* $H_i = \{\boldsymbol{x} \in \mathbb{R}^n : A_{i,\bullet}\boldsymbol{x} \leq b_i\}$



- $\boldsymbol{c}$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{c}^T\boldsymbol{x} = 0\}$)
  - *"Roll a ball downhill inside feasible region"*

  $\rightsquigarrow$ Optimal point $\boldsymbol{x}^*$ must lie on boundary!

  (assuming finite optimal objective value $z^*$)

- intersection of $n$ ~~halfspaces~~ hyperplane $H_i$ is unique point assuming nondegeneracy

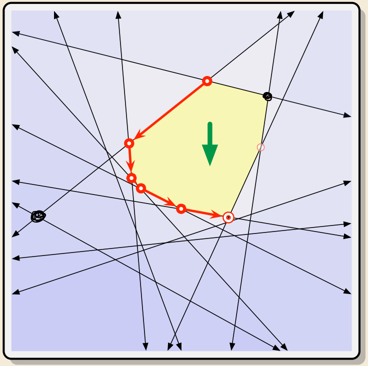
# Simplex – Geometric Intuition

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

\+ *nondegeneracy*

- constraint $A_{i,\bullet}x \leq b_i$ defines a *hyperplane*

$\rightsquigarrow$ *halfspace* $H_i = \{x \in \mathbb{R}^n : A_{i,\bullet}x \leq b_i\}$



- $c$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{x \in \mathbb{R}^n : c^T x = 0\}$)
  - *"Roll a ball downhill inside feasible region"*

  $\rightsquigarrow$ Optimal point $x^*$ must lie on boundary!

  (assuming finite optimal objective value $z^*$)

assuming nondegeneracy

- intersection of $n$ halfspaces $H_i$ is unique point

$\rightsquigarrow$ ***vertex*** $\{x_I\} = \bigcap_{i \in I} \cancel{H_i}$ (for $I \subset [m]$, $|I| = n$)

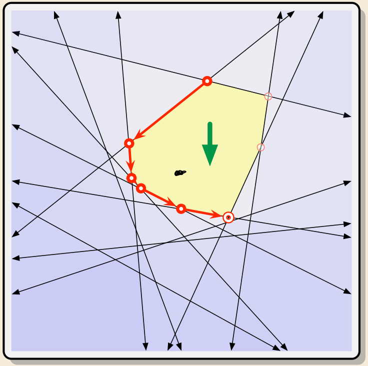
hyperplanes

# Simplex – Geometric Intuition

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax \le b \\ & x \ge 0 \end{aligned}$$
+ *nondegeneracy*

- constraint $A_{i,\bullet} x \le b_i$ defines a *hyperplane*

$\rightsquigarrow$ *halfspace* $H_i = \{x \in \mathbb{R}^n : A_{i,\bullet} x \le b_i\}$



- $c$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{x \in \mathbb{R}^n : c^T x = 0\}$)
  - *"Roll a ball downhill inside feasible region"*

  $\rightsquigarrow$ Optimal point $x^*$ must lie on boundary!

  (assuming finite optimal objective value $z^*$)

assuming nondegeneracy

- intersection of $n$ halfspaces $H_i$ is unique point

  $\rightsquigarrow$ ***vertex*** $\{x_I\} = \bigcap_{i \in I} H_i$ (for $I \subset [m]$, $|I| = n$)

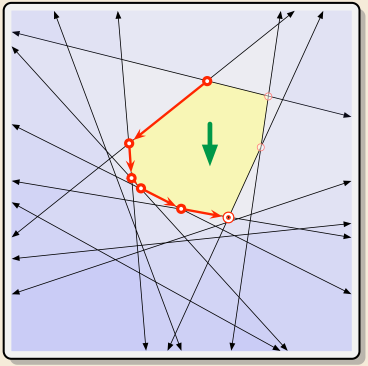
- always have $c^T x^* = c^T x_{I^*}$ for a **vertex** $x_{I^*}$

# Simplex – Geometric Intuition

$$\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax \le b \\ & x \ge 0 \end{array}$$
+ *nondegeneracy*

- constraint $A_{i,\bullet} x \le b_i$ defines a *hyperplane*

$\rightsquigarrow$ *halfspace* $H_i = \{x \in \mathbb{R}^n : A_{i,\bullet} x \le b_i\}$



- $c$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{x \in \mathbb{R}^n : c^T x = 0\}$)
  - *"Roll a ball downhill inside feasible region"*

  $\rightsquigarrow$ Optimal point $x^*$ must lie on boundary!

  (assuming finite optimal objective value $z^*$)

assuming nondegeneracy

- intersection of $n$ halfspaces $H_i$ is unique point

  $\rightsquigarrow$ ***vertex*** $\{x_I\} = \bigcap_{i \in I} H_i$ (for $I \subset [m]$, $|I| = n$)

- always have $c^T x^* = c^T x_{I^*}$ for a **vertex** $x_{I^*}$
  - "only" $\binom{m}{n}$ vertices $x_I$ (all $n$-subsets of $[m]$)

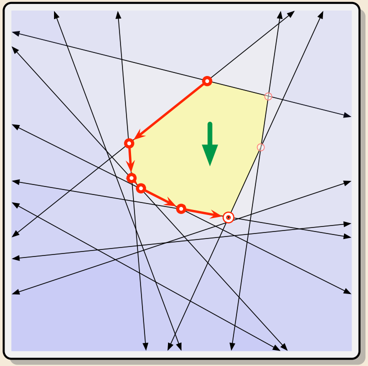
(n const $\rightsquigarrow$ polytime brute force)

# Simplex – Geometric Intuition

$$\begin{array}{rl} \min & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{array}$$
+ *nondegeneracy*

- constraint $A_{i,\bullet}\boldsymbol{x} \le b_i$ defines a *hyperplane*

  ⇝ *halfspace* $H_i = \{\boldsymbol{x} \in \mathbb{R}^n : A_{i,\bullet}\boldsymbol{x} \le b_i\}$



- $\boldsymbol{c}$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{c}^T\boldsymbol{x} = 0\}$)
  - *"Roll a ball downhill inside feasible region"*

  ⇝ Optimal point $\boldsymbol{x}^*$ must lie on boundary!

  (assuming finite optimal objective value $z^*$)

- intersection of $n$ halfspaces $H_i$ is unique point (assuming nondegeneracy)

  ⇝ ***vertex*** $\{\boldsymbol{x}_I\} = \bigcap_{i \in I} H_i$ (for $I \subset [m]$, $|I| = n$)

- always have $\boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{c}^T\boldsymbol{x}_{I^*}$ for a **vertex** $\boldsymbol{x}_{I^*}$
  - "only" $\binom{m}{n}$ vertices $\boldsymbol{x}_I$ (all $n$-subsets of $[m]$)

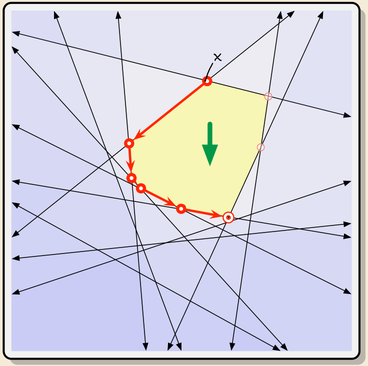
  ⇝ *Simplex algorithm*: Move to better neighbor until optimal.
  - $\boldsymbol{x}_I$ and $\boldsymbol{x}_{I'}$ neighbors if $|I \cap I'| = n - 1$

# Simplex – Geometric Intuition

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \le b \\ & x \ge 0 \end{aligned}$$

+ *nondegeneracy*

- constraint $A_{i,\bullet} x \le b_i$ defines a *hyperplane*
- ⇝ *halfspace* $H_i = \{x \in \mathbb{R}^n : A_{i,\bullet} x \le b_i\}$



- $c$ = **direction** of improvement in $\mathbb{R}^n$ (*normal vector* for hyperplane $\{x \in \mathbb{R}^n : c^T x = 0\}$)
  - *"Roll a ball downhill inside feasible region"*
  - ⇝ Optimal point $x^*$ must lie on boundary!

    (assuming finite optimal objective value $z^*$)

- intersection of $n$ halfspaces $H_i$ is unique point (assuming nondegeneracy)
  - ⇝ ***vertex*** $\{x_I\} = \bigcap_{i \in I} H_i \quad$ (for $I \subset [m], |I| = n$)
- always have $c^T x^* = c^T x_{I^*}$ for a **vertex** $x_{I^*}$
  - "only" $\binom{m}{n}$ vertices $x_I$ (all $n$-subsets of $[m]$)
  - ⇝ *Simplex algorithm*:

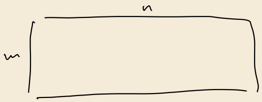
    Move to better neighbor until optimal.
  - $x_I$ and $x_{I'}$ neighbors if $|I \cap I'| = n - 1$

```
procedure simplexIteration(H = {H_1, ..., H_m}):
    if ⋂ H == ∅ return INFEASIBLE
    x := any feasible vertex
    while x is not locally optimal // c "against wall"
        // pivot towards better objective function
        if ∀ feasible neighbor vertex x' : c^T x' > c^T x
            return UNBOUNDED
        else
            x := some feasible lower neighbor of x
    return x
```

# Simplex – Linear Algebra Realization

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x \ge 0 \end{aligned}$$
+ *nondegeneracy*

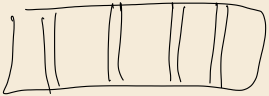
- Here use equality constraints $\rightsquigarrow$ $\underline{m \le n}$




- Assume $\operatorname{rank}(A) = m$ (nondegeneracy)
- every $J = \{j_1, \ldots, j_m\} \subseteq [n]$ corresponds to *basis* of $A$: $\{A_{\bullet,j_1}, \ldots, A_{\bullet,j_m}\}$

assuming nondegeneracy

# Simplex – Linear Algebra Realization

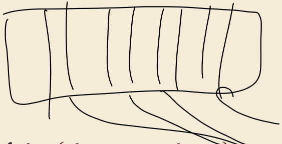


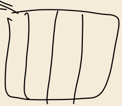
$$\begin{array}{rl} \min & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{array}$$
+ *nondegeneracy*

- Here use equality constraints $\rightsquigarrow$ $m \leq n$
- Assume $\operatorname{rank}(A) = m$ (nondegeneracy)
- every $J = \{j_1, \ldots, j_m\} \subseteq [n]$ corresponds to *basis* of $A$: $\{A_{\bullet,j_1}, \ldots, A_{\bullet,j_m}\}$
  assuming nondegeneracy

- **Notation:**
  - $\boldsymbol{x}_J = (x_{j_1}, \ldots, x_{j_m})^T$ vector of *basis variables*
  - $\boldsymbol{x}_{\bar{J}} = (x_{\bar{j}_1}, \ldots, x_{\bar{j}_{n-m}})^T$ vector of *non-basis variables* for $\bar{J} = [n] \setminus J = \{\bar{j}_1, \ldots, \bar{j}_{n-m}\}$

# Simplex – Linear Algebra Realization





$$\begin{aligned} \min\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s. t.}\ & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$
*+ nondegeneracy*

- Here use equality constraints $\rightsquigarrow$ $m \leq n$
- Assume $\operatorname{rank}(A) = m$ (nondegeneracy)
- every $J = \{j_1, \ldots, j_m\} \subseteq [n]$ corresponds to *basis* of $A$: $\{\boldsymbol{A}_{\bullet,j_1}, \ldots, \boldsymbol{A}_{\bullet,j_m}\}$

  assuming nondegeneracy

- **Notation:**
  - $\boldsymbol{x}_J = (x_{j_1}, \ldots, x_{j_m})^T$ vector of *basis variables*
  - $\boldsymbol{x}_{\bar{J}} = (x_{\bar{j}_1}, \ldots, x_{\bar{j}_{n-m}})^T$ vector of *non-basis variables* for $\bar{J} = [n] \setminus J = \{\bar{j}_1, \ldots, \bar{j}_{n-m}\}$
  - $\boldsymbol{A}_J = (\boldsymbol{A}_{\bullet,j_1}, \ldots, \boldsymbol{A}_{\bullet,j_m}) \in \mathbb{R}^{m \times m}$; similarly $\boldsymbol{A}_{\bar{J}} = (\boldsymbol{A}_{\bullet,\bar{j}_1}, \ldots, \boldsymbol{A}_{\bullet,\bar{j}_{n-m}}) \in \mathbb{R}^{(m-n)\times m}$
  - $\boldsymbol{c}_J$ and $\boldsymbol{c}_{\bar{J}}$ defined similarly

# Simplex – Linear Algebra Realization

$$\begin{array}{|rl|}\hline \min & c^T x \\ \text{s. t.} & Ax = b \\ & x \geq 0 \\ \multicolumn{2}{|l|}{+ \textit{nondegeneracy}} \\ \hline \end{array}$$

- Here use equality constraints $\rightsquigarrow$ $m \leq n$
- Assume $\operatorname{rank}(A) = m$ (nondegeneracy)
- every $J = \{j_1, \ldots, j_m\} \subseteq [n]$ corresponds to *basis* of $A$: $\{A_{\bullet,j_1}, \ldots, A_{\bullet,j_m}\}$

  assuming nondegeneracy

- **Notation:**
  - $x_J = (x_{j_1}, \ldots, x_{j_m})^T$ vector of *basis variables*
  - $x_{\bar{J}} = (x_{\bar{j}_1}, \ldots, x_{\bar{j}_{n-m}})^T$ vector of *non-basis variables* for $\bar{J} = [n] \setminus J = \{\bar{j}_1, \ldots, \bar{j}_{n-m}\}$
  - $A_J = (A_{\bullet,j_1}, \ldots, A_{\bullet,j_m}) \in \mathbb{R}^{m \times m}$; similarly $A_{\bar{J}} = (A_{\bullet,\bar{j}_1}, \ldots, A_{\bullet,\bar{j}_{n-m}}) \in \mathbb{R}^{(\cancel{m-n}) \times m}$ $n-m$
  - $c_J$ and $c_{\bar{J}}$ defined similarly $A_J$ square & full rank

$\rightsquigarrow$ We have $Ax = b \iff A_J x_J + A_{\bar{J}} x_{\bar{J}} = b \iff \boxed{x_J = A_J^{-1} b - A_J^{-1} A_{\bar{J}} x_{\bar{J}}}$

$x_J$ is uniquely determined by choosing $x_{\bar{J}}$

# Simplex – Linear Algebra Realization

$$\begin{array}{|rl|}\hline \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \\ \multicolumn{2}{|l|}{+\ \textit{nondegeneracy}} \\ \hline\end{array}$$

- Here use equality constraints $\rightsquigarrow$ $m \leq n$
- Assume $\operatorname{rank}(A) = m$ (nondegeneracy)
- every $J = \{j_1, \ldots, j_m\} \subseteq [n]$ corresponds to *basis* of $A$: $\{A_{\bullet,j_1}, \ldots, A_{\bullet,j_m}\}$

  assuming nondegeneracy

- **Notation:**
  - $x_J = (x_{j_1}, \ldots, x_{j_m})^T$ vector of *basis variables*
  - $x_{\bar{J}} = (x_{\bar{j}_1}, \ldots, x_{\bar{j}_{n-m}})^T$ vector of *non-basis variables* for $\bar{J} = [n] \setminus J = \{\bar{j}_1, \ldots, \bar{j}_{n-m}\}$
  - $A_J = (A_{\bullet,j_1}, \ldots, A_{\bullet,j_m}) \in \mathbb{R}^{m \times m}$; similarly $A_{\bar{J}} = (A_{\bullet,\bar{j}_1}, \ldots, A_{\bullet,\bar{j}_{n-m}}) \in \mathbb{R}^{(m-n) \times m}$
  - $c_J$ and $c_{\bar{J}}$ defined similarly

$\rightsquigarrow$ We have $Ax = b \iff A_J x_J + A_{\bar{J}} x_{\bar{J}} = b \iff \boxed{x_J = A_J^{-1} b - A_J^{-1} A_{\bar{J}} x_{\bar{J}}}$

$x_J$ is uniquely determined by choosing $x_{\bar{J}}$

- *basic solution* setting $x_{\bar{J}} = 0$ gives $x_J = A_J^{-1} b$ $\rightsquigarrow$ correspond to *vertices* from before
  - may or may not be a *feasible basic solution*: $x_J \geq 0$?

$\rightsquigarrow$ given $J$, can easily compute basic solution and check feasibility

# Simplex – Local Optimality Test

- basic solution: $\boxed{x_J = A_J^{-1}b - A_J^{-1}A_{\bar{J}}x_{\bar{J}}}$ and $x_{\bar{J}} = \mathbf{0}$

$$\begin{array}{rrcl} \min & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{s.t.} & A\boldsymbol{x} & = & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & \mathbf{0} \end{array}$$

+ *nondegeneracy*

# Simplex – Local Optimality Test

$$\begin{aligned} \min\ & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.}\ & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$
\+ *nondegeneracy*

- basic solution: $\boxed{\boldsymbol{x}_J = A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}}$ and $\boldsymbol{x}_{\bar{J}} = \boldsymbol{0}$
- How to locally modify basic solution without violating constraints?
  - can't change $x_{j_k}$ for $j_k \in J$ (equality constraint);
  - can't *decrease* $x_{\bar{j}_k}$ for $\bar{j}_k \in \bar{J}$ (nonnegativity);
  - ⇝ can only increase $x_{\bar{j}_k}$ by small $\delta > 0$

# Simplex – Local Optimality Test

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$
+ *nondegeneracy*

- basic solution: $\boxed{\boldsymbol{x}_J = A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}}$ and $\boldsymbol{x}_{\bar{J}} = \boldsymbol{0}$
- How to locally modify basic solution without violating constraints?
  - can't change $x_{j_k}$ for $j_k \in J$ (equality constraint);
  - can't *decrease* $x_{\bar{j}_k}$ for $\bar{j}_k \in \bar{J}$ (nonnegativity);
  - $\rightsquigarrow$ can only increase $x_{\bar{j}_k}$ by small $\delta > 0$
- rewrite cost: $\boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{c}_J\boldsymbol{x}_J + \boldsymbol{c}_{\bar{J}}^T\boldsymbol{x}_{\bar{J}}$

# Simplex – Local Optimality Test

$$\begin{aligned} \min \ & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \ & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{aligned}$$
+ *nondegeneracy*

- basic solution: $\boxed{\boldsymbol{x}_J = A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}}$ and $\boldsymbol{x}_{\bar{J}} = \boldsymbol{0}$
- How to locally modify basic solution without violating constraints?
  - can't change $x_{j_k}$ for $j_k \in J$ (equality constraint);
  - can't *decrease* $x_{\bar{j}_k}$ for $\bar{j}_k \in \bar{J}$ (nonnegativity);
  - ⇝ can only increase $x_{\bar{j}_k}$ by small $\delta > 0$
- rewrite cost:

$$\begin{aligned} \boldsymbol{c}^T\boldsymbol{x} &= \boldsymbol{c}_J\boldsymbol{x}_J + \boldsymbol{c}_{\bar{J}}^T\boldsymbol{x}_{\bar{J}} \\ &= \boldsymbol{c}_J\left(A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}\right) + \boldsymbol{c}_{\bar{J}}^T\boldsymbol{x}_{\bar{J}} \end{aligned}$$

# Simplex – Local Optimality Test

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x \geq 0 \end{aligned}$$
+ *nondegeneracy*

- basic solution: $\boxed{x_J = A_J^{-1} b - A_J^{-1} A_{\bar{J}} x_{\bar{J}}}$ and $x_{\bar{J}} = 0$
- How to locally modify basic solution without violating constraints?
  - can't change $x_{j_k}$ for $j_k \in J$ (equality constraint);
  - can't *decrease* $x_{\bar{j}_k}$ for $\bar{j}_k \in \bar{J}$ (nonnegativity);
  - ⇝ can only increase $x_{\bar{j}_k}$ by small $\delta > 0$
- rewrite cost:

$$\begin{aligned} c^T x &= c_J x_J + c_{\bar{J}}^T x_{\bar{J}} \\ &= c_J \left(A_J^{-1} b - A_J^{-1} A_{\bar{J}} x_{\bar{J}}\right) + c_{\bar{J}}^T x_{\bar{J}} \\ &= c_J A_J^{-1} b + \underbrace{\left(c_{\bar{J}}^T - c_J A_J^{-1} A_{\bar{J}}\right)}_{\tilde{c}_{\bar{J}}^T} x_{\bar{J}} \end{aligned}$$

# Simplex – Local Optimality Test

$$\begin{aligned} \min\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{aligned}$$

\+ *nondegeneracy*

- basic solution: $\boxed{\boldsymbol{x}_J = A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}}$ and $\boldsymbol{x}_{\bar{J}} = \boldsymbol{0}$
- How to locally modify basic solution without violating constraints?
  - can't change $x_{j_k}$ for $j_k \in J$ (equality constraint);
  - can't *decrease* $x_{\bar{j}_k}$ for $\bar{j}_k \in \bar{J}$ (nonnegativity);
  - ⇝ can only increase $x_{\bar{j}_k}$ by small $\delta > 0$
- rewrite cost:

$$\begin{aligned} \boldsymbol{c}^T\boldsymbol{x} &= \boldsymbol{c}_J\boldsymbol{x}_J + \boldsymbol{c}_{\bar{J}}^T\boldsymbol{x}_{\bar{J}} \\ &= \boldsymbol{c}_J\left(A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}\right) + \boldsymbol{c}_{\bar{J}}^T\boldsymbol{x}_{\bar{J}} \\ &= \boldsymbol{c}_J A_J^{-1}\boldsymbol{b} + \underbrace{\left(\boldsymbol{c}_{\bar{J}} - \boldsymbol{c}_J A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}\right)}_{\tilde{\boldsymbol{c}}_{\bar{J}}}{}^T\boldsymbol{x}_{\bar{J}} \end{aligned}$$

Convex function over a convex domain
⇝ local opt $\Longrightarrow$ global opt

⇝ **No** (local) improvement possible $\iff$ $\tilde{\boldsymbol{c}}_{\bar{J}} \ge 0$ $\iff$ current basic solution **optimal**

# Simplex – Local Optimality Test

$$\begin{aligned} \min\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.}\ & A\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \ge \boldsymbol{0} \end{aligned}$$

+ *nondegeneracy*

- basic solution: $\boxed{\boldsymbol{x}_J = A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}}$ and $\boldsymbol{x}_{\bar{J}} = \boldsymbol{0}$
- How to locally modify basic solution without violating constraints?
  - can't change $x_{j_k}$ for $j_k \in J$ (equality constraint);
  - can't *decrease* $x_{\bar{j}_k}$ for $\bar{j}_k \in \bar{J}$ (nonnegativity);
  - ⇝ can only increase $x_{\bar{j}_k}$ by small $\delta > 0$
- rewrite cost:

$$\begin{aligned} \boldsymbol{c}^T\boldsymbol{x} &= \boldsymbol{c}_J\boldsymbol{x}_J + \boldsymbol{c}_{\bar{J}}^T\boldsymbol{x}_{\bar{J}} \\ &= \boldsymbol{c}_J\left(A_J^{-1}\boldsymbol{b} - A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}\right) + \boldsymbol{c}_{\bar{J}}^T\boldsymbol{x}_{\bar{J}} \\ &= \boldsymbol{c}_J A_J^{-1}\boldsymbol{b} + \underbrace{\left(\boldsymbol{c}_{\bar{J}} - \boldsymbol{c}_J A_J^{-1}A_{\bar{J}}\boldsymbol{x}_{\bar{J}}\right)}_{\tilde{\boldsymbol{c}}_{\bar{J}}}{}^T \boldsymbol{x}_{\bar{J}} \end{aligned}$$

Convex function over a convex domain
⇝ local opt $\Longrightarrow$ global opt

⇝ **No** (local) improvement possible $\iff$ $\tilde{\boldsymbol{c}}_{\bar{J}} \ge 0$ $\iff$ current basic solution **optimal**

- Otherwise: Bring $\bar{j}_k$ with $\tilde{\boldsymbol{c}}_{\bar{j}_k} < 0$ into basis
  - This means we increase $x_{\bar{j}_k}$ as much as possible until some $x_{j_k}$ becomes $0$
  - ⇝ corresponds to moving to neighbor vertex

# Summary LP Algorithms

- **Simplex Algorithm**
  - 👍 simple and mostly combinatorial algorithm
  - 👍 easy to implement
  - 👍 usually fast in practice (in most open source solvers)

# Summary LP Algorithms

- **Simplex Algorithm**
  - simple and mostly combinatorial algorithm
  - easy to implement
  - usually fast in practice (in most open source solvers)
  - worst case running time actually **exponential**
    details depend on how better neighboring vertex is chosen (*pivoting rule*)
    but no rule known that guarantees polytime
    - but *smoothed analysis* proves: random perturbations of input yield expected polytime on any input

# Summary LP Algorithms

- **Simplex Algorithm**
  - 👍 simple and mostly combinatorial algorithm
  - 👍 easy to implement
  - 👍 usually fast in practice (in most open source solvers)
  - 👎 worst case running time actually **exponential**
    details depend on how better neighboring vertex is chosen (*pivoting rule*)
    but no rule known that guarantees polytime
    - 👍 but *smoothed analysis* proves: random perturbations of input yield expected polytime on any input

- **Alternative methods**
  - **ellipsoid method** (separation-oracle based)
  - **interior-point methods** (numeric algorithms)
  - 👍 worst case polytime
  - 👍 interior-point method fastest in practice
  - 👎 more complicated, harder to implement well