

DM545/DM871  
Linear and Integer Programming

## Linear Programming

Marco Chiarandini

Department of Mathematics & Computer Science  
University of Southern Denmark

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination

# The Diet Problem (Blending Problems)

- Select a set of foods that will satisfy a set of daily nutritional requirements at minimum cost.
- Motivated in the 1930s and 1940s by US army.
- Formulated as a **linear programming problem** by George Stigler
- (programming intended as planning not computer code)

min cost/weight

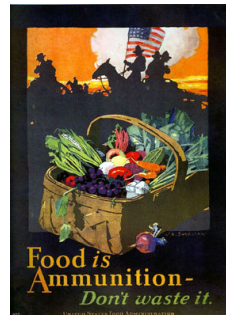
subject to nutrition requirements:

eat enough but not too much of Vitamin A

eat enough but not too much of Sodium

eat enough but not too much of Calories

...



# The Diet Problem

Suppose there are:

- 3 foods available, corn, milk, and bread, and
- there are restrictions on the number of calories (between 2000 and 2250) and the amount of Vitamin A (between 5,000 and 50,000)

Food	Cost per serving	Vitamin A	Calories
Corn	\$0.18	107	72
2% Milk	\$0.23	500	121
Wheat Bread	\$0.05	0	65

# The Mathematical Model

## Parameters (given data)

$F$  = set of foods

$N$  = set of nutrients

$a_{ij}$  = amount of nutrient  $i$  in food  $j$ ,  $\forall i \in N, \forall j \in F$

$c_j$  = cost per serving of food  $j$ ,  $\forall j \in F$

$F_{min,j}$  = minimum number of required servings of food  $j$ ,  $\forall j \in F$

$F_{max,j}$  = maximum allowable number of servings of food  $j$ ,  $\forall j \in F$

$N_{min,i}$  = minimum required level of nutrient  $i$ ,  $\forall i \in N$

$N_{max,i}$  = maximum allowable level of nutrient  $i$ ,  $\forall i \in N$

## Decision Variables

$x_j$  = number of servings of food  $j$  to purchase/consume,  $\forall j \in F$

# The Mathematical Model

**Objective Function:** Minimize the total cost of the food

$$\text{Minimize } \sum_{j \in F} c_j x_j$$

**Constraint Set 1:** For each nutrient  $i \in N$ , at least meet the minimum required level

$$\sum_{j \in F} a_{ij} x_j \geq N_{\min, i}, \quad \forall i \in N$$

**Constraint Set 2:** For each nutrient  $i \in N$ , do not exceed the maximum allowable level.

$$\sum_{j \in F} a_{ij} x_j \leq N_{\max, i}, \quad \forall i \in N$$

**Constraint Set 3:** For each food  $j \in F$ , select at least the minimum required number of servings

$$x_j \geq F_{\min, j}, \quad \forall j \in F$$

**Constraint Set 4:** For each food  $j \in F$ , do not exceed the maximum allowable number of servings.

$$x_j \leq F_{\max, j}, \quad \forall j \in F$$



# The Mathematical Model

system of equalities and inequalities

$$\min \sum_{j \in F} c_j x_j$$

$$\sum_{j \in F} a_{ij} x_j \geq N_{\min, i}, \quad \forall i \in N$$

$$\sum_{j \in F} a_{ij} x_j \leq N_{\max, i}, \quad \forall i \in N$$

$$x_j \geq F_{\min, j}, \quad \forall j \in F$$

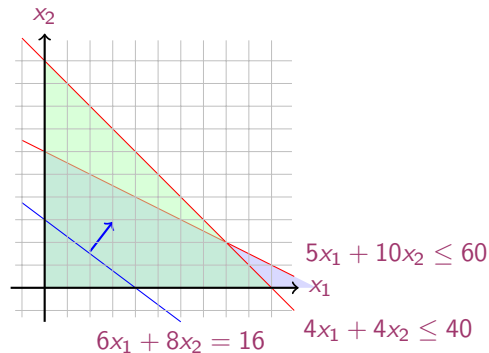
$$x_j \leq F_{\max, j}, \quad \forall j \in F$$

# Mathematical Model

Machines/Materials A and B  
Products 1 and 2

$$\begin{aligned}\max \quad & 6x_1 + 8x_2 \\ & 5x_1 + 10x_2 \leq 60 \\ & 4x_1 + 4x_2 \leq 40 \\ & x_1 \geq 0 \\ & x_2 \geq 0\end{aligned}$$

Graphical Representation:



# In Matrix Form

$$\begin{aligned}
 \max \quad & c_1x_1 + c_2x_2 + c_3x_3 + \dots + c_nx_n = z \\
 \text{s.t.} \quad & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n \leq b_1 \\
 & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n \leq b_2 \\
 & \dots \\
 & a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \dots + a_{mn}x_n \leq b_m \\
 & x_1, x_2, \dots, x_n \geq 0
 \end{aligned}$$

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$\begin{aligned}
 \max \quad & z = \mathbf{c}^T \mathbf{x} \\
 & \mathbf{Ax} \leq \mathbf{b} \\
 & \mathbf{x} \geq 0
 \end{aligned}$$

# Linear Programming

Abstract mathematical model:

Parameters, Decision Variables, Objective, Constraints  
(+ Domains & Quantifiers)

## The Syntax of a Linear Programming Problem

objective func.	$\max / \min \mathbf{c}^T \mathbf{x}$	$\mathbf{c} \in \mathbb{R}^n$
constraints	s.t. $\mathbf{Ax} \begin{matrix} \geq \\ \leq \\ = \end{matrix} \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$	$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m$ $\mathbf{x} \in \mathbb{R}^n, \mathbf{0} \in \mathbb{R}^n$

Essential features: continuity, linearity (proportionality and additivity), certainty of parameters

- Any vector  $\mathbf{x} \in \mathbb{R}^n$  satisfying all constraints is a **feasible solution**.
- Each  $\mathbf{x}^* \in \mathbb{R}^n$  that gives the best possible value for  $\mathbf{c}^T \mathbf{x}$  among all feasible  $\mathbf{x}$  is an **optimal solution** or **optimum**
- The value  $\mathbf{c}^T \mathbf{x}^*$  is the **optimum value**

# Diet Problem — History

- The linear programming model consisted of 9 equations in 77 variables
- Stigler, guessed an optimal solution using a heuristic method
- In 1947, the National Bureau of Standards used the newly developed simplex method to solve Stigler's model.  
It took 9 clerks using hand-operated desk calculators 120 man days to solve for the optimal solution
- The original instance: <http://www.gams.com/modlib/libhtml/diet.htm>

```
# diet.mod
set NUTR;
set FOOD;

param cost {FOOD} > 0;
param f_min {FOOD} >= 0;
param f_max { j in FOOD } >= f_min[j];
param n_min { NUTR } >= 0;
param n_max { i in NUTR } >= n_min[i];
param amt {NUTR,FOOD} >= 0;

var Buy { j in FOOD } >= f_min[j], <= f_max[j]

minimize total_cost: sum { j in FOOD } cost [j] * Buy[j];
subject to diet { i in NUTR } :
    n_min[i] <= sum { j in FOOD } amt[i,j] * Buy[j] <= n_max[i];
```

# AMPL Model

```
# diet.dat
data;

set NUTR := A B1 B2 C ;
set FOOD := BEEF CHK FISH HAM MCH MTL SPG TUR;

param: cost f_min f_max :=
  BEEF 3.19 0 100
  CHK 2.59 0 100
  FISH 2.29 0 100
  HAM 2.89 0 100
  MCH 1.89 0 100
  MTL 1.99 0 100
  SPG 1.99 0 100
  TUR 2.49 0 100 ;

param: n_min n_max :=
  A 700 10000
  C 700 10000
  B1 700 10000
  B2 700 10000 ;

# %
```

```
param amt (tr):
  A C B1 B2 :=
  BEEF 60 20 10 15
  CHK 8 0 20 20
  FISH 8 10 15 10
  HAM 40 40 35 10
  MCH 15 35 15 15
  MTL 70 30 15 15
  SPG 25 50 25 15
  TUR 60 20 15 10 ;
```

```
# Model diet.py
m = Model("diet")

# Create decision variables for the foods to buy
buy = {}
for f in foods:
    buy[f] = m.addVar(obj=cost[f], name=f)

# Nutrition constraints
for c in categories:
    m.addConstr(
        quicksum(nutritionValues[f,c] * buy[f] for f in foods) <= maxNutrition[c], name=c+'max')
    m.addConstr(
        quicksum(nutritionValues[f,c] * buy[f] for f in foods) >= minNutrition[c], name=c+'min')

# Solve
m.optimize()
```



```
from gurobipy import *

categories, minNutrition, maxNutrition = multidict({
    'calories': [1800, 2200],
    'protein': [91, GRB.INFINITY],
    'fat': [0, 65],
    'sodium': [0, 1779] })

foods, cost = multidict({
    'hamburger': 2.49,
    'chicken': 2.89,
    'hot dog': 1.50,
    'fries': 1.89,
    'macaroni': 2.09,
    'pizza': 1.99,
    'salad': 2.49,
    'milk': 0.89,
    'ice cream': 1.59 })
```

```
# Nutrition values for the foods
nutritionValues = {
    ('hamburger', 'calories'): 410,
    ('hamburger', 'protein'): 24,
    ('hamburger', 'fat'): 26,
    ('hamburger', 'sodium'): 730,
    ('chicken', 'calories'): 420,
    ('chicken', 'protein'): 32,
    ('chicken', 'fat'): 10,
    ('chicken', 'sodium'): 1190,
    ('hot dog', 'calories'): 560,
    ('hot dog', 'protein'): 20,
    ('hot dog', 'fat'): 32,
    ('hot dog', 'sodium'): 1800,
    ('fries', 'calories'): 380,
    ('fries', 'protein'): 4,
    ('fries', 'fat'): 19,
    ('fries', 'sodium'): 270,
    ('macaroni', 'calories'): 320,
    ('macaroni', 'protein'): 12,
    ('macaroni', 'fat'): 10,
    ('macaroni', 'sodium'): 930,
    ('pizza', 'calories'): 320,
    ('pizza', 'protein'): 15,
    ...
}
```

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination

# History of Linear Programming (LP)

## System of linear equations

↪ It is impossible to find out who knew what when first.

Just two “references”:

- Egyptians and Babylonians considered about 2000 B.C. the solution of special linear equations. But, of course, they described examples and did not describe the methods in "today's style".
- What we call “Gaussian elimination” today has been explicitly described in Chinese “Nine Books of Arithmetic” which is a compendium written in the period 2010 B.C. to A.D. 9, but the methods were probably known long before that.
- Gauss, by the way, never described “Gaussian elimination”. He just used it and stated that the linear equations he used can be solved “per eliminationem vulgarem”

# History of Linear Programming (LP)

- Origins date back to Newton, Leibnitz, Lagrange, etc.
- In 1827, Fourier described a variable elimination method for **systems of linear inequalities**, today often called Fourier-Motzkin elimination (Motzkin, 1937). It can be turned into an LP solver but inefficient.
- In 1932, Leontief (1905-1999) Input-Output model to represent interdependencies between branches of a national economy (1976 Nobel prize)
- In 1939, Kantorovich (1912-1986): Foundations of linear programming (Nobel prize in economics with Koopmans on LP, 1975) on Optimal use of scarce resources: foundation and economic interpretation of LP
- The math subfield of **Linear Programming** was created by George Dantzig, John von Neumann (Princeton), and Leonid Kantorovich in the 1940s.
- In 1947, Dantzig (1914-2005) invented the **(primal) simplex algorithm** working for the US Air Force at the Pentagon. (program=plan)

# History of LP (cntd)

- In 1954, Lemke: dual simplex algorithm,
- In 1954, Dantzig and Orchard Hays: revised simplex algorithm
- In 1970, Victor Klee and George Minty created an example that showed that the classical simplex algorithm has exponential worst-case behavior.
- In 1979, L. Khachain found a new **efficient** algorithm for linear programming. It was terribly slow. (Ellipsoid method)
- In 1984, Karmarkar discovered yet another new **efficient** algorithm for linear programming. It proved to be a strong competitor for the simplex method. (Interior point method)

# History of Optimization

- In 1951, Nonlinear Programming began with the Karush-Kuhn-Tucker Conditions
- In 1952, Commercial Applications and Software began
- In 1950s, Network Flow Theory began with the work of Ford and Fulkerson.
- In 1955, Stochastic Programming began
- In 1958, Integer Programming began by R. E. Gomory.
- In 1962, Complementary Pivot Theory

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination

# Fourier Motzkin elimination method

Has  $Ax \leq b$  a solution? (Assumption:  $A \in \mathbb{Q}^{m \times n}$ ,  $b \in \mathbb{Q}^n$ )

Idea:

1. transform the system into another by eliminating some variables such that the two systems have the same solutions over the remaining variables.
2. reduce to a system of constant inequalities that can be easily decided

Let  $x_r$  be the variable to eliminate

Let  $M = \{1 \dots m\}$  indices of the constraints

For a variable  $j$  let's partition the rows of the matrix in

$$N = \{i \in M \mid a_{ij} < 0\}$$

$$Z = \{i \in M \mid a_{ij} = 0\}$$

$$P = \{i \in M \mid a_{ij} > 0\}$$



$$\left\{ \begin{array}{ll} x_r \geq b'_{ir} - \sum_{k=1}^{r-1} a'_{ik} x_k, & a_{ir} < 0 \\ x_r \leq b'_{ir} - \sum_{k=1}^{r-1} a'_{ik} x_k, & a_{ir} > 0 \\ \text{all other constraints} & i \in Z \end{array} \right. \quad \left\{ \begin{array}{ll} x_r \geq A_i(x_1, \dots, x_{r-1}), & i \in N \\ x_r \leq B_i(x_1, \dots, x_{r-1}), & i \in P \\ \text{all other constraints} & i \in Z \end{array} \right.$$

Hence the original system is equivalent to

$$\left\{ \begin{array}{l} \max\{A_i(x_1, \dots, x_{r-1}), i \in N\} \leq x_r \leq \min\{B_i(x_1, \dots, x_{r-1}), i \in P\} \\ \text{all other constraints} \quad i \in Z \end{array} \right.$$

which is equivalent to

$$\left\{ \begin{array}{ll} A_i(x_1, \dots, x_{r-1}) \leq B_j(x_1, \dots, x_{r-1}) & i \in N, j \in P \\ \text{all other constraints} & i \in Z \end{array} \right.$$

we eliminated  $x_r$  but:

$$\left\{ \begin{array}{l} |N| \cdot |P| \text{ inequalities} \\ |Z| \text{ inequalities} \end{array} \right.$$

after  $d$  iterations if  $|P| = |N| = m/2$  exponential growth:  $(1/4^d)(m/2)^{2^d}$

# Example

$$-7x_1 + 6x_2 \leq 25$$

$$x_1 - 5x_2 \leq 1$$

$$x_1 \leq 7$$

$$-x_1 + 2x_2 \leq 12$$

$$-x_1 - 3x_2 \leq 1$$

$$2x_1 - x_2 \leq 10$$

$x_2$  variable to eliminate

$$N = \{2, 5, 6\}, Z = \{3\}, P = \{1, 4\}$$

$$|Z \cup (N \times P)| = 7 \text{ constraints}$$

By adding one variable and one inequality, Fourier-Motzkin elimination can be turned into an LP solver.

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination

# Definitions

- $\mathbb{R}$ : set of real numbers  
 $\mathbb{N} = \{1, 2, 3, 4, \dots\}$ : set of natural numbers (positive integers)  
 $\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$ : set of all integers  
 $\mathbb{Q} = \{p/q \mid p, q \in \mathbb{Z}, q \neq 0\}$ : set of rational numbers

- column vector and matrices  
 scalar product:  $\mathbf{y}^T \mathbf{x} = \sum_{i=1}^n y_i x_i$

- linear combination

$$\begin{aligned} \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k &\in \mathbb{R}^n \\ \boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_k]^T &\in \mathbb{R}^k \end{aligned} \quad \mathbf{x} = \lambda_1 \mathbf{v}_1 + \dots + \lambda_k \mathbf{v}_k = \sum_{i=1}^k \lambda_i \mathbf{v}_i$$

moreover:

$$\boldsymbol{\lambda} \geq \mathbf{0}$$

$$\boldsymbol{\lambda}^T \mathbf{1} = 1$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \text{ and } \boldsymbol{\lambda}^T \mathbf{1} = 1$$

conic combination

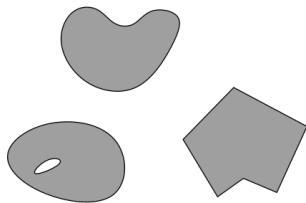
affine combination

convex combination

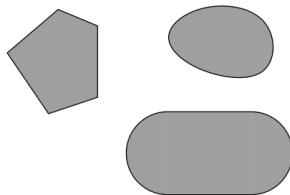
$$\left( \sum_{i=1}^k \lambda_i = 1 \right)$$

# Definitions

- set  $S$  is **linear (affine) independent** if no element of it can be expressed as linear combination of the others  
Eg:  $S \subseteq \mathbb{R}^n \implies \max n \text{ lin. indep. } (\max n+1 \text{ aff. indep.})$
- **convex set**: if  $\mathbf{x}, \mathbf{y} \in S$  and  $0 \leq \lambda \leq 1$  then  $\lambda \mathbf{x} + (1 - \lambda) \mathbf{y} \in S$



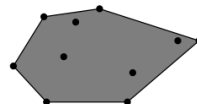
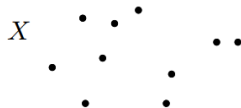
nonconvex



convex

- **convex function** if its epigraph  $\{(x, y) \in \mathbb{R}^2 : y \geq f(x)\}$  is a convex set or  $f : X \rightarrow \mathbb{R}$ , or if  $\forall x, y \in X, \lambda \in [0, 1]$  it holds that  $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$

- For a set of points  $S \subseteq \mathbb{R}^n$ 
  - $\text{lin}(S)$  linear hull (span)
  - $\text{cone}(S)$  conic hull
  - $\text{aff}(S)$  affine hull
  - $\text{conv}(S)$  convex hull



the convex hull of  $X$

$$\text{conv}(X) = \left\{ \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 + \dots + \lambda_n \mathbf{x}_n \mid \mathbf{x}_i \in X, \lambda_1, \dots, \lambda_n \geq 0 \text{ and } \sum_i \lambda_i = 1 \right\}$$

# Definitions

- **rank** of a matrix for columns (= for rows)  
if  $(m, n)$ -matrix has rank  $= \min\{m, n\}$  then the matrix is full rank  
if  $(n, n)$ -matrix is full rank then it is regular and admits an inverse

- $G \subseteq \mathbb{R}^n$  is an **hyperplane** if  $\exists \mathbf{a} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  and  $\alpha \in \mathbb{R}$ :

$$G = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^T \mathbf{x} = \alpha\}$$

- $H \subseteq \mathbb{R}^n$  is an **halfspace** if  $\exists \mathbf{a} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  and  $\alpha \in \mathbb{R}$ :

$$H = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^T \mathbf{x} \leq \alpha\}$$

( $\mathbf{a}^T \mathbf{x} = \alpha$  is a supporting hyperplane of  $H$ )

# Definitions

- a set  $S \subset \mathbb{R}^n$  is a **polyhedron** if  $\exists m \in \mathbb{Z}^+, A \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m$ :

$$P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \leq \mathbf{b}\} = \bigcap_{i=1}^m \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}_{i,\cdot} \mathbf{x} \leq b_i\}$$

i.e., a polyhedron  $P \neq \mathbb{R}^n$  is determined by finitely many halfspaces

- a polyhedron  $P$  is a **polytope** if it is bounded:  $\exists B \in \mathbb{R}, B > 0$ :

$$P \subseteq \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq B\}$$

- A set of vectors is a **polytope** if it is the convex hull of finitely many vectors.



# Definitions

- If  $A$  and  $b$  are made of rational numbers,  $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$  is a rational polyhedron
- General optimization problem:  $\max\{\varphi(x) \mid x \in F\}$ ,  $F$  is feasible region for  $x$
- Note: if  $F$  is open, eg,  $x < 5$  then:  $\sup\{x \mid x < 5\}$   
 sumpreum: least element of  $\mathbb{R}$  greater or equal than any element in  $F$

# Definitions

- The inequality denoted by  $(\mathbf{a}, \alpha)$  is called a **valid inequality for  $P$**  if  $\mathbf{a}\mathbf{x} \leq \alpha, \forall \mathbf{x} \in P$ .  
Note that  $(\mathbf{a}, \alpha)$  is a valid inequality if and only if  $P$  lies in the half-space  $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}\mathbf{x} \leq \alpha\}$ .
- A **face** of  $P$  is  $F = \{\mathbf{x} \in P \mid \mathbf{a}\mathbf{x} = \alpha\}$  where  $(\mathbf{a}, \alpha)$  is a valid inequality for  $P$ . Hence, it is the intersection of  $P$  with the hyperplane of a valid inequality. It is said to be **proper** if  $F \neq \emptyset$  and  $F \neq P$ .
- If  $F \neq \emptyset$  we say that it **supports  $P$** .  
If  $\mathbf{c}$  is a non zero vector for which  $\delta = \max\{\mathbf{c}^T \mathbf{x} \mid \mathbf{x} \in P\}$  is finite, then the set  $\{\mathbf{x} \mid \mathbf{c}^T \mathbf{x} = \delta\}$  is called **supporting hyperplane**.
- A point  $\mathbf{x}$  for which  $\{\mathbf{x}\}$  is a face is called a **vertex** of  $P$  and also a **basic solution** of  $A\mathbf{x} \leq \mathbf{b}$  (**0 dim** face)
- A **facet** is a maximal face distinct from  $P$   
 $\mathbf{c}\mathbf{x} \leq \mathbf{d}$  is facet defining if  $\mathbf{c}\mathbf{x} = \mathbf{d}$  is a supporting hyperplane of  $P$  of  $n - 1$  dim

# Linear Programming Problem

**Input:** a matrix  $A \in \mathbb{R}^{m \times n}$  and column vectors  $\mathbf{b} \in \mathbb{R}^m$ ,  $\mathbf{c} \in \mathbb{R}^n$

**Task:**

1. decide that  $\{\mathbf{x} \in \mathbb{R}^n; A\mathbf{x} \leq \mathbf{b}\}$  is empty (prob. infeasible), or
2. find a column vector  $\mathbf{x} \in \mathbb{R}^n$  such that  $A\mathbf{x} \leq \mathbf{b}$  and  $\mathbf{c}^T \mathbf{x}$  is max, or
3. decide that for all  $\alpha \in \mathbb{R}$  there is an  $\mathbf{x} \in \mathbb{R}^n$  with  $A\mathbf{x} \leq \mathbf{b}$  and  $\mathbf{c}^T \mathbf{x} > \alpha$  (prob. unbounded)

1.  $F = \emptyset$
2.  $F \neq \emptyset$  and  $\exists$  solution
  1. one solution
  2. infinite solutions
3.  $F \neq \emptyset$  and  $\nexists$  solution

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination

# Fundamental Theorem of LP

## Theorem (Fundamental Theorem of Linear Programming)

Given:

$$\min\{\mathbf{c}^T \mathbf{x} \mid \mathbf{x} \in P\} \text{ where } P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \leq \mathbf{b}\}$$

If  $P$  is a bounded polyhedron and not empty and  $\mathbf{x}^*$  is an optimal solution to the problem, then:

- $\mathbf{x}^*$  is an extreme point (vertex) of  $P$ , or
- $\mathbf{x}^*$  lies on a face  $F \subset P$  of optimal solutions



Proof idea:

- assume  $\mathbf{x}^*$  not a vertex of  $P$  then  $\exists$  a ball around it still in  $P$ . Show that a point in the ball has better cost
- if  $\mathbf{x}^*$  is not a vertex then it is a convex combination of vertices. Show that all points are also optimal.

## Implications:

- the optimal solution is at the intersection of supporting hyperplanes.
- hence finitely many possibilities
- solution method: write all inequalities as equalities and solve all  $\binom{m}{n}$  systems of linear equalities ( $n$  # variables,  $m$  # equality constraints)
- for each point we then need to check if feasible and if best in cost.
- each system is solved by Gaussian elimination
- Stirling approximation:

$$\binom{2m}{m} \approx \frac{4^m}{\sqrt{\pi m}} \text{ as } m \rightarrow \infty$$

1. find a solution that is at the intersection of some  $n$  hyperplanes
2. try systematically to produce the other points by exchanging one hyperplane with another
3. check optimality, proof provided by duality theory

# Outline

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination



# Gaussian Elimination

## 1. Forward elimination

reduces the system to row echelon form by elementary row operations

- multiply a row by a non-zero constant
- interchange two rows
- add a multiple of one row to another

(or LU decomposition)

## 2. Back substitution (or reduced row echelon form - RREF)

## Example

$$\begin{aligned} 2x + y - z &= 8 & (R1) \\ -3x - y + 2z &= -11 & (R2) \\ -2x + y + 2z &= -3 & (R3) \end{aligned}$$

$$\begin{array}{c|cccc|c} \hline & 2 & 1 & -1 & 8 \\ \hline R1 & 2 & 1 & -1 & 8 \\ R2 & -3 & -1 & 2 & -11 \\ R3 & -2 & 1 & 2 & -3 \\ \hline \end{array}$$

$$\begin{aligned} 2x + y - z &= 8 & (R1) \\ + \frac{1}{2}y + \frac{1}{2}z &= 1 & (R2) \\ + 2y + 1z &= 5 & (R3) \end{aligned}$$

$$\begin{array}{c|cccc|c} \hline & 1 & 1/2 & -1/2 & 4 \\ \hline R1' = 1/2 R1 & 1 & 1/2 & -1/2 & 4 \\ R2' = R2 + 3/2 R1 & 0 & 1/2 & 1/2 & 1 \\ R3' = R3 + R1 & 0 & 2 & 1 & 5 \\ \hline \end{array}$$

$$\begin{aligned} 2x + y - z &= 8 & (R1) \\ + \frac{1}{2}y + \frac{1}{2}z &= 1 & (R2) \\ - z &= 1 & (R3) \end{aligned}$$

$$\begin{array}{c|cccc|c} \hline & 1 & 1/2 & -1/2 & 4 \\ \hline R1' = R1 & 1 & 1/2 & -1/2 & 4 \\ R2' = 2 R2 & 0 & 1 & 1 & 2 \\ R3' = R3 - 4 R2 & 0 & 0 & -1 & 1 \\ \hline \end{array}$$

$$\begin{aligned} 2x + y - z &= 8 & (R1) \\ + \frac{1}{2}y + \frac{1}{2}z &= 1 & (R2) \\ - z &= 1 & (R3) \end{aligned}$$

$$\begin{array}{c|cccc|c} \hline & 1 & 1/2 & 0 & 7/2 \\ \hline R1' = R1 - 1/2 R3 & 1 & 1/2 & 0 & 7/2 \\ R2' = R2 + R3 & 0 & 1 & 0 & 3 \\ R3' = -R3 & 0 & 0 & 1 & -1 \\ \hline \end{array}$$

$$\begin{aligned} x &= 2 & (R1) \\ y &= 3 & (R2) \\ z &= -1 & (R3) \end{aligned}$$

$$\begin{array}{c|cccc|c} \hline & 1 & 0 & 0 & 2 \\ \hline R1' = R1 - 1/2 R2 & 1 & 0 & 0 & 2 \\ R2' = R2 & 0 & 1 & 0 & 3 \\ R3' = R3 & 0 & 0 & 1 & -1 \\ \hline \end{array} \Rightarrow \begin{aligned} x &= 2 \\ y &= 3 \\ z &= -1 \end{aligned}$$

# LU Factorization

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix}$$

$$A\mathbf{x} = \mathbf{b}$$

$$\mathbf{x} = A^{-1}\mathbf{b}$$

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$$A = PLU$$

$$\mathbf{x} = A^{-1}\mathbf{b} = U^{-1}L^{-1}P^T\mathbf{b}$$

$$\mathbf{z}_1 = P^T\mathbf{b}, \quad \mathbf{z}_2 = L^{-1}\mathbf{z}_1, \quad \mathbf{x} = U^{-1}\mathbf{z}_2$$

```
In [1]: import scipy as sc
```

```
In [2]: A = sc.array([[2,1,-1],[-3,-1,2],[-2,1,2]])
```

```
In [3]: from scipy import linalg as sl
```

```
In [4]: P,L,U = sl.lu(A)
```

```
In [5]: print(P,L,U)
```

```
[[0. 0. 1.]
```

```
 [1. 0. 0.]
```

```
 [0. 1. 0.]]
```

```
[[ 1. 0. 0.]
```

```
 [ 0.66666667 1. 0. ]
```

```
 [-0.66666667 0.2 1. ]]
```

```
[[ -3. -1. 2.]
```

```
 [ 0. 1.66666667 0.66666667]
```

```
 [ 0. 0. 0.2 ]]
```

Polynomial time  $O(n^2m)$  but needs to guarantee that all the numbers during the run can be represented by polynomially bounded bits

# Summary

## 1. Introduction

Diet Problem

## 2. Solving LP Problems

Fourier-Motzkin method

## 3. Preliminaries

Fundamental Theorem of LP

Gaussian Elimination