# SS 2015

# Efficient Algorithms
# and Data Structures II

Harald Räcke

Fakultät für Informatik
TU München

`http://www14.in.tum.de/lehre/2015SS/ea/`

Summer Term 2015

---

# Part I

# Organizational Matters

---

# Part I

# Organizational Matters

- ▶ Modul: IN2004
- ▶ Name: "Efficient Algorithms and Data Structures II"
  "Effiziente Algorithmen und Datenstrukturen II"
- ▶ ECTS: 8 Credit points
- ▶ Lectures:
    - ▶ 4 SWS
      Mon 12:15-13:45 (Room 00.13.009A)
      Fri 12:15-13:45 (Room 00.13.009A)
- ▶ Webpage: `http://www14.in.tum.de/lehre/2015SS/ea/`

---

# The Lecturer

- ▶ Harald Räcke
- ▶ Email: raecke@in.tum.de
- ▶ Room: 03.09.044
- ▶ Office hours: (per appointment)

## Tutorials

- Tutor:
  - Chintan Shah
  - chintan.shah@tum.de
  - Room: 03.09.059
  - per appointment
- Room: 03.11.018
- Time: Tue 16:15–17:45

## Assessment

- In order to pass the module you need to pass an exam.

- Exam:
  - 3 hours
  - Date will be announced shortly.
  - There are no resources allowed, apart from a hand-written piece of paper (A4).
  - Answers should be given in English, but German is also accepted.

## Assessment

- Assignment Sheets:
  - An assignment sheet is usually made available on Monday on the module webpage.
  - Solutions have to be handed in in the following week before the lecture on Monday.
  - You can hand in your solutions by putting them in the right folder in front of room 03.09.020.
  - Solutions have to be given in English.
  - Solutions will be discussed in the subsequent tutorial on Tuesday.
  - The first one will be out on Monday, 20 April.

## 1 Contents

Part 1: Linear Programming

Part 2: Approximation Algorithms

## 2 Literatur

- 📄 V. Chvatal:
  *Linear Programming,*
  Freeman, 1983
- 📄 R. Seidel:
  *Skript Optimierung,* 1996
- 📄 D. Bertsimas and J.N. Tsitsiklis:
  *Introduction to Linear Optimization,*
  Athena Scientific, 1997
- 📄 Vijay V. Vazirani:
  *Approximation Algorithms,*
  Springer 2001

- 📄 David P. Williamson and David B. Shmoys:
  *The Design of Approximation Algorithms,*
  Cambridge University Press 2011
- 📄 G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi:
  *Complexity and Approximation,*
  Springer, 1999

# Part II

# Linear Programming

## Brewery Problem

**Brewery brews ale and beer.**

- ▶ Production limited by supply of corn, hops and barley malt
- ▶ Recipes for ale and beer require different amounts of resources

|  | Corn (kg) | Hops (kg) | Malt (kg) | Profit (€) |
|---|---|---|---|---|
| ale (barrel) | 5 | 4 | 35 | 13 |
| beer (barrel) | 15 | 4 | 20 | 23 |
| supply | 480 | 160 | 1190 | |

## Brewery Problem

| | Corn (kg) | Hops (kg) | Malt (kg) | Profit (€) |
|---|---|---|---|---|
| ale (barrel) | 5 | 4 | 35 | 13 |
| beer (barrel) | 15 | 4 | 20 | 23 |
| supply | 480 | 160 | 1190 | |

**How can brewer maximize profits?**

- only brew ale: 34 barrels of ale $\implies$ 442 €
- only brew beer: 32 barrels of beer $\implies$ 736 €
- 7.5 barrels ale, 29.5 barrels beer $\implies$ 776 €
- 12 barrels ale, 28 barrels beer $\implies$ 800 €

---

## Brewery Problem

**Linear Program**

- Introduce variables $a$ and $b$ that define how much ale and beer to produce.
- Choose the variables in such a way that the objective function (profit) is maximized.
- Make sure that no constraints (due to limited supply) are violated.

$$
\begin{aligned}
\max \quad & 13a + 23b \\
\text{s.t.} \quad & 5a + 15b \leq 480 \\
& 4a + 4b \leq 160 \\
& 35a + 20b \leq 1190 \\
& a, b \geq 0
\end{aligned}
$$

---

## Standard Form LPs

**LP in standard form:**

- input: numbers $a_{ij}$, $c_j$, $b_i$
- output: numbers $x_j$
- $n$ = #decision variables, $m$ = #constraints
- maximize linear objective function subject to linear (in)equalities

$$
\begin{aligned}
\max \quad & \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j = b_i \quad 1 \leq i \leq m \\
& x_j \geq 0 \quad 1 \leq j \leq n
\end{aligned}
$$

$$
\begin{aligned}
\max \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0
\end{aligned}
$$

---

## Standard Form LPs

**Original LP**

$$
\begin{aligned}
\max \quad & 13a + 23b \\
\text{s.t.} \quad & 5a + 15b \leq 480 \\
& 4a + 4b \leq 160 \\
& 35a + 20b \leq 1190 \\
& a, b \geq 0
\end{aligned}
$$

**Standard Form**
Add a slack variable to every constraint.

$$
\begin{aligned}
\max \quad & 13a + 23b \\
\text{s.t.} \quad & 5a + 15b + s_c = 480 \\
& 4a + 4b + s_h = 160 \\
& 35a + 20b + s_m = 1190 \\
& a, b, s_c, s_h, s_m \geq 0
\end{aligned}
$$

## Standard Form LPs

There are different standard forms:

standard form

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad Ax &= b \\ x &\geq 0 \end{aligned}$$

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

standard
maximization form

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad Ax &\leq b \\ x &\geq 0 \end{aligned}$$

standard
minimization form

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

---

## Standard Form LPs

It is easy to transform variants of LPs into (any) standard form:

▶ **less or equal to equality:**

$$a - 3b + 5c \leq 12 \implies \begin{array}{c} a - 3b + 5c + s = 12 \\ s \geq 0 \end{array}$$

▶ **greater or equal to equality:**

$$a - 3b + 5c \geq 12 \implies \begin{array}{c} a - 3b + 5c - s = 12 \\ s \geq 0 \end{array}$$

▶ **min to max:**

$$\min a - 3b + 5c \implies \max -a + 3b - 5c$$

---

## Standard Form LPs

It is easy to transform variants of LPs into (any) standard form:

▶ **equality to less or equal:**

$$a - 3b + 5c = 12 \implies \begin{array}{c} a - 3b + 5c \leq 12 \\ -a + 3b - 5c \leq -12 \end{array}$$

▶ **equality to greater or equal:**

$$a - 3b + 5c = 12 \implies \begin{array}{c} a - 3b + 5c \geq 12 \\ -a + 3b - 5c \geq -12 \end{array}$$

▶ **unrestricted to nonnegative:**

$$x \text{ unrestricted} \implies x = x^+ - x^-, x^+ \geq 0, x^- \geq 0$$

---

## Standard Form LPs

**Observations:**

▶ a linear program does not contain $x^2$, $\cos(x)$, etc.

▶ transformations between standard forms can be done efficiently and only change the size of the LP by a small constant factor

▶ for the standard minimization or maximization LPs we could include the nonnegativity constraints into the set of ordinary constraints; this is of course not possible for the standard form

## Fundamental Questions

**Definition 1 (Linear Programming Problem (LP))**

Let $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, $\alpha \in \mathbb{Q}$. Does there exist $x \in \mathbb{Q}^n$ s.t. $Ax = b$, $x \geq 0$, $c^T x \geq \alpha$?

**Questions**:
- ▸ Is LP in NP?
- ▸ Is LP in co-NP?
- ▸ Is LP in P?

**Input size**:
- ▸ $n$ number of variables, $m$ constraints, $L$ number of bits to encode the input

---

## Geometry of Linear Programming



---

## Geometry of Linear Programming



Regardless of the objective function an optimum solution occurs at a vertex (Ecke).

---
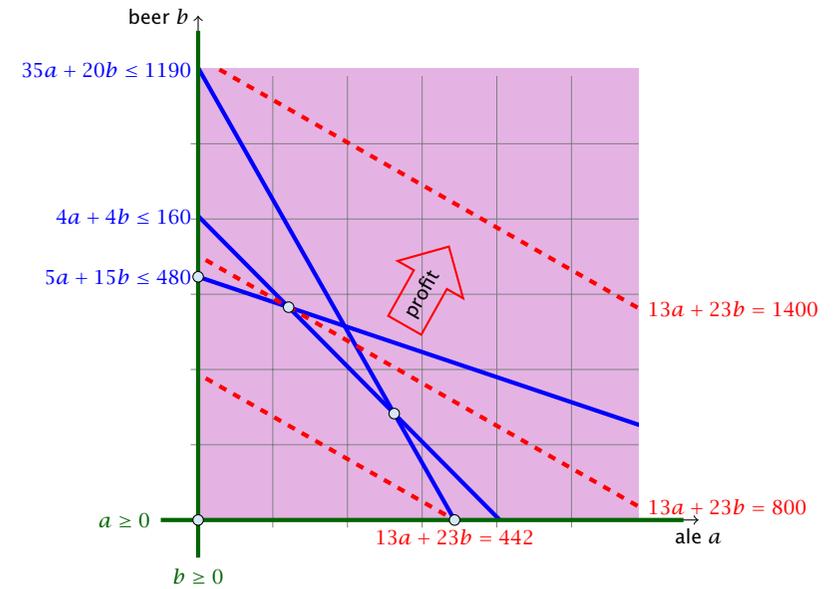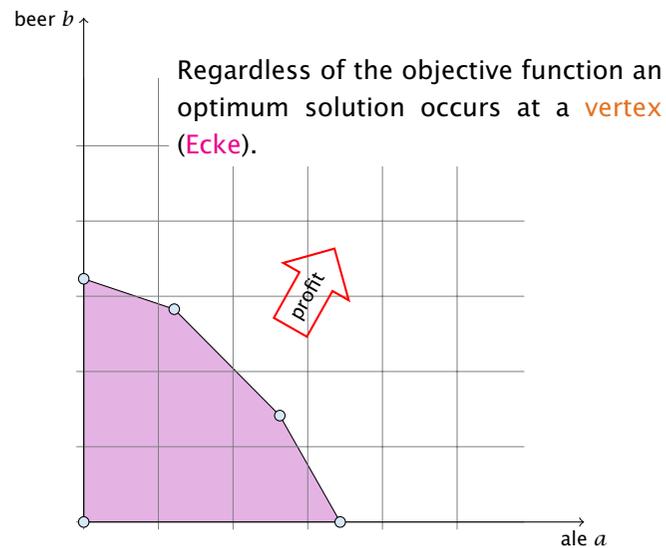
## Definitions

Let for a Linear Program in standard form
$P = \{x \mid Ax = b, x \geq 0\}$.

- ▸ $P$ is called the feasible region (Lösungsraum) of the LP.
- ▸ A point $x \in P$ is called a feasible point (gültige Lösung).
- ▸ If $P \neq \emptyset$ then the LP is called feasible (erfüllbar). Otherwise, it is called infeasible (unerfüllbar).
- ▸ An LP is bounded (beschränkt) if it is feasible and
  - ▸ $c^T x < \infty$ for all $x \in P$ (for maximization problems)
  - ▸ $c^T x > -\infty$ for all $x \in P$ (for minimization problems)

**Definition 2**

Given points $x, y \in \mathbb{R}^n$, a point $z \in \mathbb{R}^n$ is a convex combination of $x$ and $y$ if

$$z = \lambda x + (1 - \lambda) y$$

for some $\lambda \in [0, 1]$.

**Definition 3**

A set $X \subseteq \mathbb{R}^n$ is convex if the convex combination of any two poins in $X$ is also in $X$.

**Definition 4**

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$ we have

$$f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y)$$

**Lemma 5**

*If $P \subseteq \mathbb{R}^n$, and $f : \mathbb{R}^n \to \mathbb{R}$ convex than also*

$$Q = \{x \in P \mid f(x) \leq t\}$$

**Definition 6**

The dimension of a set $X \subseteq \mathbb{R}^n$ is the dimension of the vector space generated by vectors of the form $(x - y)$ with $x, y \in X$.

**Definition 7**

A set $H \subseteq \mathbb{R}^n$ is a hyperplane if $H = \{x \mid a^T x = b\}$, for $a \neq 0$.

**Definition 8**

A set $H' \subseteq \mathbb{R}^n$ is a (closed) halfspace if $H = \{x \mid a^T x \leq b\}$, for $a \neq 0$.

# Definitions

**Definition 9**

A polytop is a set $P \subseteq \mathbb{R}^n$ that is the convex hull of a finite set of points, i.e., $P = \text{conv}(X)$ where

$$\text{conv}(X) = \left\{ \sum_{i=1}^{\ell} \lambda_i x_i \mid \ell \in \mathbb{N}, x_1, \ldots, x_\ell \in X, \lambda_i \geq 0, \sum_i \lambda_i = 1 \right\}$$

and $|X| = c$.

## Definitions

### Definition 10
A polyhedron is a set $P \subseteq \mathbb{R}^n$ that can be represented as the intersection of finitely many half-spaces $\{H(a_1, b_1), \ldots, H(a_m, b_m)\}$, where

$$H(a_i, b_i) = \{x \in \mathbb{R}^n \mid a_i x \leq b_i\} \ .$$

### Definition 11
A polyhedron $P$ is bounded if there exists $B$ s.t. $\|x\|_2 \leq B$ for all $x \in P$.

## Definitions

### Theorem 12
*P is a bounded polyhedron iff P is a polytop.*

### Definition 13
Let $P \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. The hyperplane

$$H(a, b) = \{x \in \mathbb{R}^n \mid ax = b\}$$

is a supporting hyperplane of $P$ if $\max\{ax \mid x \in P\} = b$.

### Definition 14
Let $P \subseteq \mathbb{R}^n$. $F$ is a face of $P$ if $F = P$ or $F = P \cap H$ for some supporting hyperplane $H$.

### Definition 15
Let $P \subseteq \mathbb{R}^n$.
- a face $v$ is a vertex of $P$ if $\{v\}$ is a face of $P$.
- a face $e$ is an edge of $P$ if $e$ is a face and $\dim(e) = 1$.
- a face $F$ is a facet of $P$ if $F$ is a face and $\dim(F) = \dim(P) - 1$.

**Equivalent definition for vertex:**

### Definition 16
Given polyhedron $P$. A point $x \in P$ is a vertex if $\exists c \in \mathbb{R}^n$ such that $c^T x < c^T y$, for all $y \in P$.

### Definition 17
Given polyhedron $P$. A point $x \in P$ is an extreme point if $\nexists a, b \neq x$, $a, b \in P$, with $\lambda a + (1 - \lambda)b = x$ for $\lambda \in [0, 1]$.

### Lemma 18
*A vertex is also an extreme point.*

**Observation**

The feasible region of an LP is a Polyhedron.

## Convex Sets

### Theorem 19

*If there exists an optimal solution to an LP (in standard form) then there exists an optimum solution that is an extreme point.*

### Proof

▸ suppose $x$ is optimal solution that is not extreme point

▸ there exists direction $d \neq 0$ such that $x \pm d \in P$

▸ $Ad = 0$ because $A(x \pm d) = b$

▸ Wlog. assume $c^T d \geq 0$ (by taking either $d$ or $-d$)

▸ Consider $x + \lambda d$, $\lambda > 0$

## Convex Sets

**Case 1.** [$\exists j$ s.t. $d_j < 0$]

▸ increase $\lambda$ to $\lambda'$ until first component of $x + \lambda d$ hits 0

▸ $x + \lambda' d$ is feasible. Since $A(x + \lambda' d) = b$ and $x + \lambda' d \geq 0$

▸ $x + \lambda' d$ has one more zero-component ($d_k = 0$ for $x_k = 0$ as $x \pm d \in P$)

▸ $c^T x' = c^T (x + \lambda' d) = c^T x + \lambda' c^T d \geq c^T x$

**Case 2.** [$d_j \geq 0$ for all $j$ and $c^T d > 0$]

▸ $x + \lambda d$ is feasible for all $\lambda \geq 0$ since $A(x + \lambda d) = b$ and $x + \lambda d \geq x \geq 0$

▸ as $\lambda \to \infty$, $c^T (x + \lambda d) \to \infty$ as $c^T d > 0$

## Algebraic View



An extreme point in $\mathbb{R}^d$ is uniquely defined by $d$ linearly independent equations.
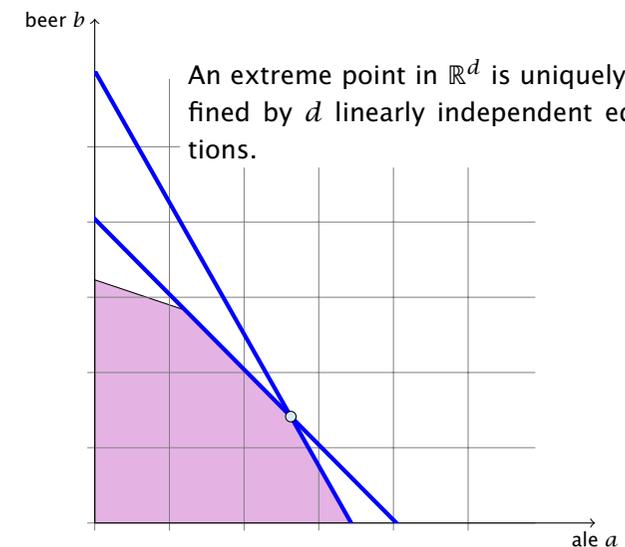
**Notation**

Suppose $B \subseteq \{1 \ldots n\}$ is a set of column-indices. Define $A_B$ as the subset of columns of $A$ indexed by $B$.

**Theorem 20**

Let $P = \{x \mid Ax = b, x \geq 0\}$. For $x \in P$, define $B = \{j \mid x_j > 0\}$.
Then $x$ is extreme point **iff** $A_B$ has linearly independent columns.

---

**Theorem 20**

Let $P = \{x \mid Ax = b, x \geq 0\}$. For $x \in P$, define $B = \{j \mid x_j > 0\}$.
Then $x$ is extreme point **iff** $A_B$ has linearly independent columns.

**Proof ($\Leftarrow$)**

- assume $x$ is not extreme point
- there exists direction $d$ s.t. $x \pm d \in P$
- $Ad = 0$ because $A(x \pm d) = b$
- define $B' = \{j \mid d_j \neq 0\}$
- $A_{B'}$ has linearly dependent columns as $Ad = 0$
- $d_j = 0$ for all $j$ with $x_j = 0$ as $x \pm d \geq 0$
- Hence, $B' \subseteq B$, $A_{B'}$ is sub-matrix of $A_B$

---

**Theorem 20**

Let $P = \{x \mid Ax = b, x \geq 0\}$. For $x \in P$, define $B = \{j \mid x_j > 0\}$.
Then $x$ is extreme point **iff** $A_B$ has linearly independent columns.

**Proof ($\Rightarrow$)**

- assume $A_B$ has linearly dependent columns
- there exists $d \neq 0$ such that $A_B d = 0$
- extend $d$ to $\mathbb{R}^n$ by adding $0$-components
- now, $Ad = 0$ and $d_j = 0$ whenever $x_j = 0$
- for sufficiently small $\lambda$ we have $x \pm \lambda d \in P$
- hence, $x$ is not extreme point

---

**Theorem 20**

Let $P = \{x \mid Ax = b, x \geq 0\}$. For $x \in P$, define $B = \{j \mid x_j > 0\}$.
If $A_B$ has linearly independent columns then $x$ is a vertex of $P$.

- define $c_j = \begin{cases} 0 & j \in B \\ 1 & j \notin B \end{cases}$
- then $c^T x = 0$ and $c^T y \geq 0$ for $y \in P$
- assume $c^T y = 0$; then $y_j = 0$ for all $j \notin B$
- $b = Ay = A_B y_B = Ax = A_B x_B$ gives that $A_B(x_B - Y_B) = 0$;
- this means that $x_B = y_B$ since $A_B$ has linearly independent columns
- we get $y = x$
- hence, $x$ is a vertex of $P$

**Observation**

For an LP we can assume wlog. that the matrix $A$ has full row-rank. This means $\text{rank}(A) = m$.

- ▶ assume that $\text{rank}(A) < m$
- ▶ assume wlog. that the first row $A_1$ lies in the span of the other rows $A_2, \ldots, A_m$; this means

$$A_1 = \sum_{i=2}^{m} \lambda_i \cdot A_i, \text{ for suitable } \lambda_i$$

- **C1** if now $b_1 = \sum_{i=2}^{m} \lambda_i \cdot b_i$ then for all $x$ with $A_i x = b_i$ we also have $A_1 x = b_1$; hence the first constraint is superfluous
- **C2** if $b_1 \neq \sum_{i=2}^{m} \lambda_i \cdot b_i$ then the LP is infeasible, since for all $x$ that fulfill constraints $A_2, \ldots, A_m$ we have

$$A_1 x = \sum_{i=2}^{m} \lambda_i \cdot A_i x = \sum_{i=2}^{m} \lambda_i \cdot b_i \neq b_1$$

From now on we will always assume that the constraint matrix of a standard form LP has full row rank.

**Theorem 21**

*Given $P = \{x \mid Ax = b, x \geq 0\}$. $x$ is extreme point iff there exists $B \subseteq \{1, \ldots, n\}$ with $|B| = m$ and*

- ▶ $A_B$ is non-singular
- ▶ $x_B = A_B^{-1} b \geq 0$
- ▶ $x_N = 0$

where $N = \{1, \ldots, n\} \setminus B$.

**Proof**

Take $B = \{j \mid x_j > 0\}$ and augment with linearly independent columns until $|B| = m$; always possible since $\text{rank}(A) = m$.

## Basic Feasible Solutions

$x \in \mathbb{R}^n$ is called basic solution (Basislösung) if $Ax = b$ and $\text{rank}(A_J) = |J|$ where $J = \{j \mid x_j \neq 0\}$;

$x$ is a basic **feasible** solution (gültige Basislösung) if in addition $x \geq 0$.

A basis (Basis) is an index set $B \subseteq \{1, \ldots, n\}$ with $\text{rank}(A_B) = m$ and $|B| = m$.

$x \in \mathbb{R}^n$ with $A_B x = b$ and $x_j = 0$ for all $j \notin B$ is the basic solution associated to basis B (die zu $B$ assoziierte Basislösung)

## Basic Feasible Solutions

A BFS fulfills the $m$ equality constraints.

In addition, at least $n - m$ of the $x_i$'s are zero. The corresponding non-negativity constraint is fulfilled with equality.

**Fact:**
In a BFS at least $n$ constraints are fulfilled with equality.

## Basic Feasible Solutions

**Definition 22**
For a general LP ($\min\{c^T x \mid Ax \geq b\}$) with $n$ variables a point $x$ is a basic feasible solution if $x$ is feasible and there exist $n$ (linearly independent) constraints that are tight.

## Algebraic View



$$\begin{aligned} \max\ & 13a + 23b \\ \text{s.t.}\quad & 5a + 15b + s_c && = 480 \\ & 4a + 4b && + s_h && = 160 \\ & 35a + 20b && && + s_m = 1190 \\ & a\ ,\quad b\ ,\ s_c\ ,\ s_h\ ,\ s_m \geq 0 \end{aligned}$$
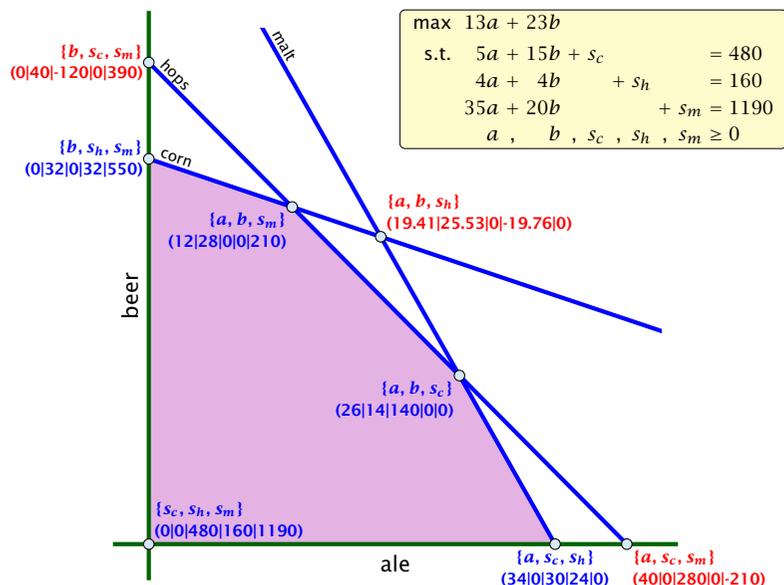
## Fundamental Questions

**Linear Programming Problem (LP)**
Let $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, $\alpha \in \mathbb{Q}$. Does there exist $x \in \mathbb{Q}^n$ s.t. $Ax = b$, $x \geq 0$, $c^T x \geq \alpha$?

**Questions**:
▶ Is LP in NP? yes!
▶ Is LP in co-NP?
▶ Is LP in P?

**Proof**:
▶ Given a basis $B$ we can compute the associated basis solution by calculating $A_B^{-1} b$ in polynomial time; then we can also compute the profit.

**Observation**
We can compute an optimal solution to a linear program in time
$\mathcal{O}\left(\binom{n}{m} \cdot \text{poly}(n, m)\right)$.

- ▶ there are only $\binom{n}{m}$ different bases.
- ▶ compute the profit of each of them and take the maximum

# 4 Simplex Algorithm

Enumerating all basic feasible solutions (BFS), in order to find the optimum is slow.

**Simplex Algorithm** [George Dantzig 1947]
Move from BFS to adjacent BFS, without decreasing objective function.

Two BFSs are called adjacent if the bases just differ in one variable.

# 4 Simplex Algorithm

$$
\begin{aligned}
\max \ & 13a + 23b \\
\text{s.t.} \quad & 5a + 15b + s_c && = 480 \\
& 4a + 4b && + s_h && = 160 \\
& 35a + 20b && + s_m = 1190 \\
& a, \quad b, \ s_c, \ s_h, \ s_m \geq 0
\end{aligned}
$$

$$
\begin{aligned}
\max \ Z \\
& 13a + 23b && - Z = 0 \\
& 5a + 15b + s_c && = 480 \\
& 4a + 4b + s_h && = 160 \\
& 35a + 20b + s_m && = 1190 \\
& a, \quad b, \ s_c, \ s_h, \ s_m && \geq 0
\end{aligned}
$$

basis = $\{s_c, s_h, s_m\}$
$A = B = 0$
$Z = 0$

$s_c = 480$
$s_h = 160$
$s_m = 1190$

# Pivoting Step

$$
\begin{aligned}
\max \ Z \\
& 13a + 23b && - Z = 0 \\
& 5a + 15b + s_c && = 480 \\
& 4a + 4b + s_h && = 160 \\
& 35a + 20b + s_m && = 1190 \\
& a, \quad b, \ s_c, \ s_h, \ s_m && \geq 0
\end{aligned}
$$

basis = $\{s_c, s_h, s_m\}$
$a = b = 0$
$Z = 0$

$s_c = 480$
$s_h = 160$
$s_m = 1190$

- ▶ choose variable to bring into the basis
- ▶ chosen variable should have positive coefficient in objective function
- ▶ apply min-ratio test to find out by how much the variable can be increased
- ▶ pivot on row found by min-ratio test
- ▶ the existing basis variable in this row leaves the basis

max $Z$

$$
\begin{aligned}
13a + 23\mathbf{b} \qquad\qquad\qquad - Z &= 0 \\
5a + 15\mathbf{b} + s_c \qquad\qquad &= 480 \\
4a + 4\mathbf{b} \qquad + s_h \qquad &= 160 \\
35a + 20\mathbf{b} \qquad\qquad + s_m &= 1190 \\
a \,, \quad \mathbf{b} \,, \; s_c \,, \; s_h \,, \; s_m \;\; &\geq 0
\end{aligned}
$$

basis = $\{s_c, s_h, s_m\}$
$a = b = 0$
$Z = 0$

$s_c = 480$
$s_h = 160$
$s_m = 1190$

- ▶ Choose variable with coefficient $\geq 0$ as entering variable.
- ▶ If we keep $a = 0$ and increase $b$ from $0$ to $\theta > 0$ s.t. all constraints ($Ax = b, x \geq 0$) are still fulfilled the objective value $Z$ will strictly increase.
- ▶ For maintaining $Ax = b$ we need e.g. to set $s_c = 480 - 15\theta$.
- ▶ Choosing $\theta = \min\{480/15, 160/4, 1190/20\}$ ensures that in the new solution one current basic variable becomes $0$, and no variable goes negative.
- ▶ The basic variable in the row that gives $\min\{480/15, 160/4, 1190/20\}$ becomes the leaving variable.

---

max $Z$

$$
\begin{aligned}
13a + 23b \qquad\qquad\qquad - Z &= 0 \\
5a + 15b + s_c \qquad\qquad &= 480 \\
4a + 4b \qquad + s_h \qquad &= 160 \\
35a + 20b \qquad\qquad + s_m &= 1190 \\
a \,, \quad b \,, \; s_c \,, \; s_h \,, \; s_m \;\; &\geq 0
\end{aligned}
$$

basis = $\{s_c, s_h, s_m\}$
$a = b = 0$
$Z = 0$

$s_c = 480$
$s_h = 160$
$s_m = 1190$

Substitute $b = \frac{1}{15}(480 - 5a - s_c)$.

max $Z$

$$
\begin{aligned}
\tfrac{16}{3}a \qquad - \tfrac{23}{15}s_c \qquad\qquad\qquad - Z &= -736 \\
\tfrac{1}{3}a + b + \tfrac{1}{15}s_c \qquad\qquad &= 32 \\
\tfrac{8}{3}a \qquad - \tfrac{4}{15}s_c + s_h \qquad &= 32 \\
\tfrac{85}{3}a \qquad - \tfrac{4}{3}s_c \qquad + s_m &= 550 \\
a \,, b \,, \quad s_c \,, \; s_h \,, \; s_m \;\; &\geq 0
\end{aligned}
$$

basis = $\{b, s_h, s_m\}$
$a = s_c = 0$
$Z = 736$

$b = 32$
$s_h = 32$
$s_m = 550$

---

max $Z$

$$
\begin{aligned}
\tfrac{16}{3}\mathbf{a} \qquad - \tfrac{23}{15}s_c \qquad\qquad\qquad - Z &= -736 \\
\tfrac{1}{3}\mathbf{a} + b + \tfrac{1}{15}s_c \qquad\qquad &= 32 \\
\tfrac{8}{3}\mathbf{a} \qquad - \tfrac{4}{15}s_c + s_h \qquad &= 32 \\
\tfrac{85}{3}\mathbf{a} \qquad - \tfrac{4}{3}s_c \qquad + s_m &= 550 \\
\mathbf{a} \,, b \,, \quad s_c \,, \; s_h \,, \; s_m \;\; &\geq 0
\end{aligned}
$$

basis = $\{b, s_h, s_m\}$
$a = s_c = 0$
$Z = 736$

$b = 32$
$s_h = 32$
$s_m = 550$

Choose variable $a$ to bring into basis.
Computing $\min\{3 \cdot 32, 3\cdot 32/8, 3\cdot 550/85\}$ means pivot on line 2.
Substitute $a = \frac{3}{8}(32 + \frac{4}{15}s_c - s_h)$.

max $Z$

$$
\begin{aligned}
- s_c - 2s_h \qquad - Z &= -800 \\
b + \tfrac{1}{10}s_c - \tfrac{1}{8}s_h \qquad &= 28 \\
a \qquad - \tfrac{1}{10}s_c + \tfrac{3}{8}s_h \qquad &= 12 \\
\tfrac{3}{2}s_c - \tfrac{85}{8}s_h + s_m &= 210 \\
a \,, b \,, \quad s_c \,, \quad s_h \,, \; s_m \;\; &\geq 0
\end{aligned}
$$

basis = $\{a, b, s_m\}$
$s_c = s_h = 0$
$Z = 800$

$b = 28$
$a = 12$
$s_m = 210$

---

# 4 Simplex Algorithm

Pivoting stops when all coefficients in the objective function are non-positive.

**Solution is optimal:**

- ▶ any feasible solution satisfies all equations in the tableaux
- ▶ in particular: $Z = 800 - s_c - 2s_h$, $s_c \geq 0, s_h \geq 0$
- ▶ hence optimum solution value is at most $800$
- ▶ the current solution has value $800$

## Matrix View

Let our linear program be

$$
\begin{array}{rcrcl}
c_B^T x_B & + & c_N^T x_N & = & Z \\
A_B x_B & + & A_N x_N & = & b \\
x_B & , & x_N & \geq & 0
\end{array}
$$

The simplex tableaux for basis $B$ is

$$
\begin{array}{rcrcl}
 & & (c_N^T - c_B^T A_B^{-1} A_N) x_N & = & Z - c_B^T A_B^{-1} b \\
I x_B & + & A_B^{-1} A_N x_N & = & A_B^{-1} b \\
x_B & , & x_N & \geq & 0
\end{array}
$$

The BFS is given by $x_N = 0, x_B = A_B^{-1} b$.

If $(c_N^T - c_B^T A_B^{-1} A_N) \leq 0$ we know that we have an optimum solution.

---

## Geometric View of Pivoting



$$
\begin{array}{llr}
\max & 13a + 23b & \\
\text{s.t.} & 5a + 15b + s_c & = 480 \\
& 4a + 4b + s_h & = 160 \\
& 35a + 20b + s_m & = 1190 \\
& a , \ b , \ s_c , \ s_h , \ s_m \geq 0 &
\end{array}
$$

---
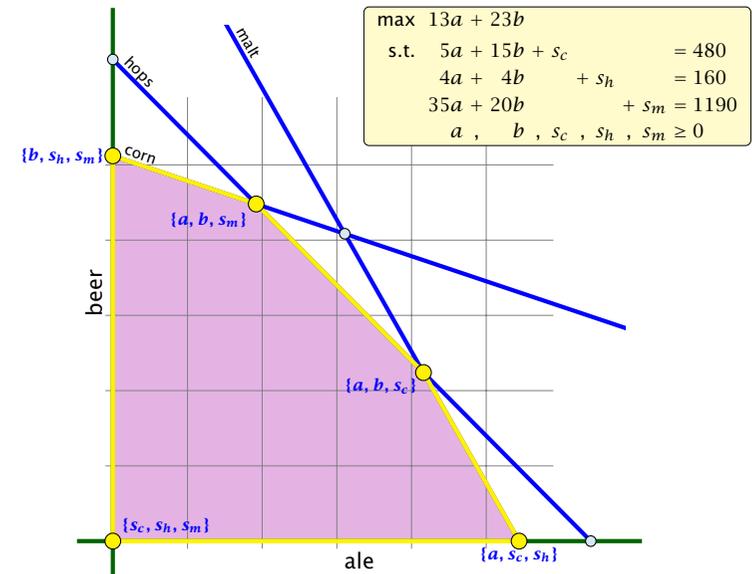
## Algebraic Definition of Pivoting

- ▶ Given basis $B$ with BFS $x^*$.
- ▶ Choose index $j \notin B$ in order to increase $x_j^*$ from $0$ to $\theta > 0$.
  - ▶ Other non-basis variables should stay at $0$.
  - ▶ Basis variables change to maintain feasibility.
- ▶ Go from $x^*$ to $x^* + \theta \cdot d$.

**Requirements for $d$:**

- ▶ $d_j = 1$ (normalization)
- ▶ $d_\ell = 0, \ell \notin B, \ell \neq j$
- ▶ $A(x^* + \theta d) = b$ must hold. Hence $Ad = 0$.
- ▶ Altogether: $A_B d_B + A_{*j} = Ad = 0$, which gives $d_B = -A_B^{-1} A_{*j}$.

---

## Algebraic Definition of Pivoting

**Definition 23 ($j$-th basis direction)**

Let $B$ be a basis, and let $j \notin B$. The vector $d$ with $d_j = 1$ and $d_\ell = 0, \ell \notin B, \ell \neq j$ and $d_B = -A_B^{-1} A_{*j}$ is called the *$j$-th basis direction* for $B$.

Going from $x^*$ to $x^* + \theta \cdot d$ the objective function changes by

$$
\theta \cdot c^T d = \theta(c_j - c_B^T A_B^{-1} A_{*j})
$$

## Algebraic Definition of Pivoting

### Definition 24 (Reduced Cost)

For a basis $B$ the value

$$\tilde{c}_j = c_j - c_B^T A_B^{-1} A_{*j}$$

is called the reduced cost for variable $x_j$.

Note that this is defined for every $j$. If $j \in B$ then the above term is $0$.

## Algebraic Definition of Pivoting

Let our linear program be

$$
\begin{array}{rcrcl}
c_B^T x_B & + & c_N^T x_N & = & Z \\
A_B x_B & + & A_N x_N & = & b \\
x_B & , & x_N & \geq & 0
\end{array}
$$

The simplex tableaux for basis $B$ is

$$
\begin{array}{rcrcl}
& & (c_N^T - c_B^T A_B^{-1} A_N) x_N & = & Z - c_B^T A_B^{-1} b \\
I x_B & + & A_B^{-1} A_N x_N & = & A_B^{-1} b \\
x_B & , & x_N & \geq & 0
\end{array}
$$

The BFS is given by $x_N = 0, x_B = A_B^{-1} b$.

If $(c_N^T - c_B^T A_B^{-1} A_N) \leq 0$ we know that we have an optimum solution.

## 4 Simplex Algorithm

**Questions:**

▶ What happens if the min ratio test fails to give us a value $\theta$ by which we can safely increase the entering variable?

▶ How do we find the initial basic feasible solution?

▶ Is there always a basis $B$ such that

$$(c_N^T - c_B^T A_B^{-1} A_N) \leq 0 \ ?$$

Then we can terminate because we know that the solution is optimal.

▶ If yes how do we make sure that we reach such a basis?

## Min Ratio Test

The min ratio test computes a value $\theta \geq 0$ such that after setting the entering variable to $\theta$ the leaving variable becomes $0$ and all other variables stay non-negative.

For this, one computes $b_i / A_{ie}$ for all constraints $i$ and calculates the minimum positive value.

What does it mean that the ratio $b_i / A_{ie}$ (and hence $A_{ie}$) is negative for a constraint?

This means that the corresponding basic variable will increase if we increase $b$. Hence, there is no danger of this basic variable becoming negative

What happens if **all** $b_i / A_{ie}$ are negative? Then we do not have a leaving variable. Then the LP is unbounded!

## Termination

The objective function does not decrease during one iteration of the simplex-algorithm.

Does it always increase?

## Termination

The objective function may not increase!

Because a variable $x_\ell$ with $\ell \in B$ is already $0$.

The set of inequalities is degenerate (also the basis is degenerate).

### Definition 25 (Degeneracy)
A BFS $x^*$ is called degenerate if the set $J = \{j \mid x_j^* > 0\}$ fulfills $|J| < m$.

It is possible that the algorithm cycles, i.e., it cycles through a sequence of different bases without ever terminating. Happens, very rarely in practise.

## Non Degenerate Example



$$\max \ 13a + 23b$$
$$\text{s.t.} \quad 5a + 15b + s_c \qquad\qquad = 480$$
$$4a + 4b \qquad + s_h \qquad = 160$$
$$35a + 20b \qquad\qquad + s_m = 1190$$
$$a \ , \quad b \ , \ s_c \ , \ s_h \ , \ s_m \geq 0$$

## Degenerate Example



$$\max \qquad 13a + 23b$$
$$\text{s.t.} \qquad 5a + 15b + s_c \qquad\qquad = 480$$
$$80/17 \cdot a + 4b \qquad + s_h \qquad = 160$$
$$35a + 20b \qquad\qquad + s_m = 1190$$
$$a \ , \quad b \ , \ s_c \ , \ s_h \ , \ s_m \geq 0$$
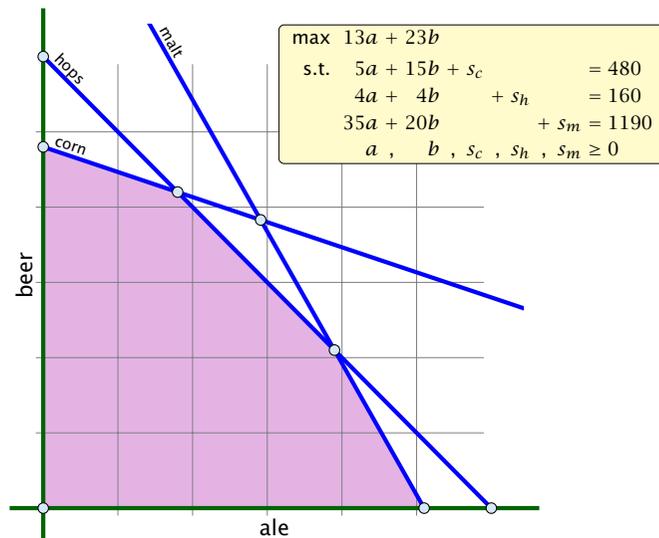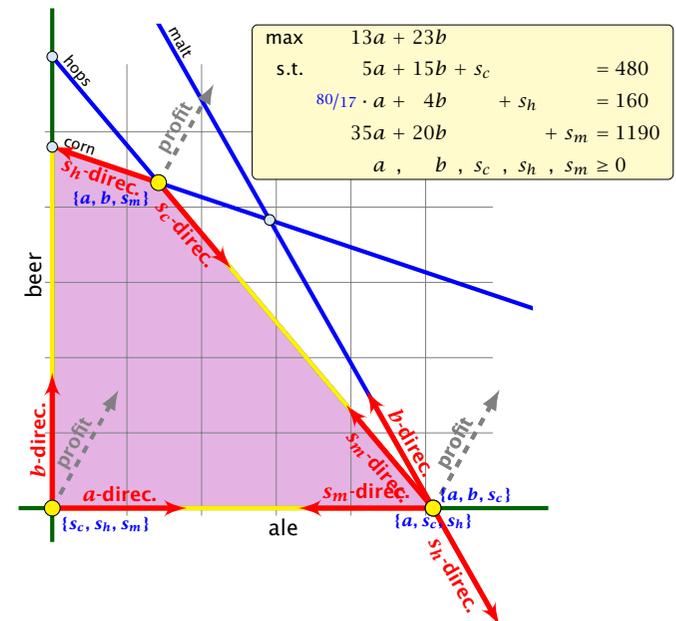
## Summary: How to choose pivot-elements

- ▸ We can choose a column $e$ as an entering variable if $\tilde{c}_e > 0$ ($\tilde{c}_e$ is reduced cost for $x_e$).
- ▸ The standard choice is the column that maximizes $\tilde{c}_e$.
- ▸ If $A_{ie} \leq 0$ for all $i \in \{1, \ldots, m\}$ then the maximum is not bounded.
- ▸ Otw. choose a leaving variable $\ell$ such that $b_\ell / A_{\ell e}$ is minimal among all variables $i$ with $A_{ie} > 0$.
- ▸ If several variables have minimum $b_\ell / A_{\ell e}$ you reach a degenerate basis.
- ▸ Depending on the choice of $\ell$ it may happen that the algorithm runs into a cycle where it does not escape from a degenerate vertex.

## Termination

**What do we have so far?**
Suppose we are given an initial feasible solution to an LP. If the LP is non-degenerate then Simplex will terminate.

Note that we either terminate because the min-ratio test fails and we can conclude that the LP is unbounded, or we terminate because the vector of reduced cost is non-positive. In the latter case we have an optimum solution.

**How do we come up with an initial solution?**

- ▸ $Ax \leq b, x \geq 0$, and $b \geq 0$.
- ▸ The standard slack from for this problem is $Ax + Is = b, x \geq 0, s \geq 0$, where $s$ denotes the vector of slack variables.
- ▸ Then $s = b, x = 0$ is a basic feasible solution (how?).
- ▸ We directly can start the simplex algorithm.

How do we find an initial basic feasible solution for an arbitrary problem?

## Two phase algorithm

Suppose we want to maximize $c^T x$ s.t. $Ax = b, x \geq 0$.

1. Multiply all rows with $b_i < 0$ by $-1$.
2. maximize $-\sum_i v_i$ s.t. $Ax + Iv = b$, $x \geq 0$, $v \geq 0$ using Simplex. $x = 0$, $v = b$ is initial feasible.
3. If $\sum_i v_i > 0$ then the original problem is infeasible.
4. Otw. you have $x \geq 0$ with $Ax = b$.
5. From this you can get basic feasible solution.
6. Now you can start the Simplex for the original problem.

## Optimality

### Lemma 26

*Let $B$ be a basis and $x^*$ a BFS corresponding to basis $B$. $\tilde{c} \leq 0$ implies that $x^*$ is an optimum solution to the LP.*

## Duality

**How do we get an upper bound to a maximization LP?**

$$
\begin{array}{rrcrcl}
\max & 13a & + & 23b & & \\
\text{s.t.} & 5a & + & 15b & \leq & 480 \\
& 4a & + & 4b & \leq & 160 \\
& 35a & + & 20b & \leq & 1190 \\
& & & a, b & \geq & 0
\end{array}
$$

Note that a lower bound is easy to derive. Every choice of $a, b \geq 0$ gives us a lower bound (e.g. $a = 12, b = 28$ gives us a lower bound of $800$).

If you take a conic combination of the rows (multiply the $i$-th row with $y_i \geq 0$) such that $\sum_i y_i a_{ij} \geq c_j$ then $\sum_i y_i b_i$ will be an upper bound.

## Duality

### Definition 27

Let $z = \max\{c^T x \mid Ax \leq b, x \geq 0\}$ be a linear program $P$ (called the primal linear program).

The linear program $D$ defined by

$$w = \min\{b^T y \mid A^T y \geq c, y \geq 0\}$$

is called the dual problem.

## Duality

### Lemma 28

*The dual of the dual problem is the primal problem.*

**Proof:**
- $w = \min\{b^T y \mid A^T y \geq c, y \geq 0\}$
- $w = -\max\{-b^T y \mid -A^T y \leq -c, y \geq 0\}$

The dual problem is
- $z = -\min\{-c^T x \mid -Ax \geq -b, x \geq 0\}$
- $z = \max\{c^T x \mid Ax \leq b, x \geq 0\}$

## Weak Duality

Let $z = \max\{c^T x \mid Ax \leq b, x \geq 0\}$ and
$w = \min\{b^T y \mid A^T y \geq c, y \geq 0\}$ be a primal dual pair.

$x$ is primal feasible iff $x \in \{x \mid Ax \leq b, x \geq 0\}$

$y$ is dual feasible, iff $y \in \{y \mid A^T y \geq c, y \geq 0\}$.

### Theorem 29 (Weak Duality)
*Let $\hat{x}$ be primal feasible and let $\hat{y}$ be dual feasible. Then*

$$c^T \hat{x} \leq z \leq w \leq b^T \hat{y} .$$

## Weak Duality

$A^T \hat{y} \geq c \Rightarrow \hat{x}^T A^T \hat{y} \geq \hat{x}^T c \; (\hat{x} \geq 0)$

$A\hat{x} \leq b \Rightarrow y^T A\hat{x} \leq \hat{y}^T b \; (\hat{y} \geq 0)$

This gives
$$c^T \hat{x} \leq \hat{y}^T A\hat{x} \leq b^T \hat{y} .$$

Since, there exists primal feasible $\hat{x}$ with $c^T \hat{x} = z$, and dual feasible $\hat{y}$ with $b^T y = w$ we get $z \leq w$.

If $P$ is unbounded then $D$ is infeasible.

The following linear programs form a primal dual pair:

$$z = \max\{c^T x \mid Ax = b, x \geq 0\}$$
$$w = \min\{b^T y \mid A^T y \geq c\}$$

This means for computing the dual of a standard form LP, we do not have non-negativity constraints for the dual variables.

## 5.2 Simplex and Duality

The following linear programs form a primal dual pair:

$$z = \max\{c^T x \mid Ax = b, x \geq 0\}$$
$$w = \min\{b^T y \mid A^T y \geq c\}$$

This means for computing the dual of a standard form LP, we do not have non-negativity constraints for the dual variables.

## Proof

**Primal:**

$$\max\{c^T x \mid Ax = b, x \geq 0\}$$
$$= \max\{c^T x \mid Ax \leq b, -Ax \leq -b, x \geq 0\}$$
$$= \max\left\{c^T x \mid \begin{bmatrix} A \\ -A \end{bmatrix} x \leq \begin{bmatrix} b \\ -b \end{bmatrix}, x \geq 0\right\}$$

**Dual:**

$$\min\{[b^T \ -b^T] y \mid [A^T \ -A^T] y \geq c, y \geq 0\}$$
$$= \min\left\{[b^T \ -b^T] \cdot \begin{bmatrix} y^+ \\ y^- \end{bmatrix} \, \middle| \, [A^T \ -A^T] \cdot \begin{bmatrix} y^+ \\ y^- \end{bmatrix} \geq c, y^- \geq 0, y^+ \geq 0\right\}$$
$$= \min\left\{b^T \cdot (y^+ - y^-) \, \middle| \, A^T \cdot (y^+ - y^-) \geq c, y^- \geq 0, y^+ \geq 0\right\}$$
$$= \min\left\{b^T y' \, \middle| \, A^T y' \geq c\right\}$$

## Proof of Optimality Criterion for Simplex

Suppose that we have a basic feasible solution with reduced cost

$$\tilde{c} = c^T - c_B^T A_B^{-1} A \leq 0$$

This is equivalent to $A^T (A_B^{-1})^T c_B \geq c$

$y^* = (A_B^{-1})^T c_B$ is solution to the dual $\min\{b^T y \mid A^T y \geq c\}$.

$$b^T y^* = (Ax^*)^T y^* = (A_B x_B^*)^T y^*$$
$$= (A_B x_B^*)^T (A_B^{-1})^T c_B = (x_B^*)^T A_B^T (A_B^{-1})^T c_B$$
$$= c^T x^*$$

Hence, the solution is optimal.

## ?
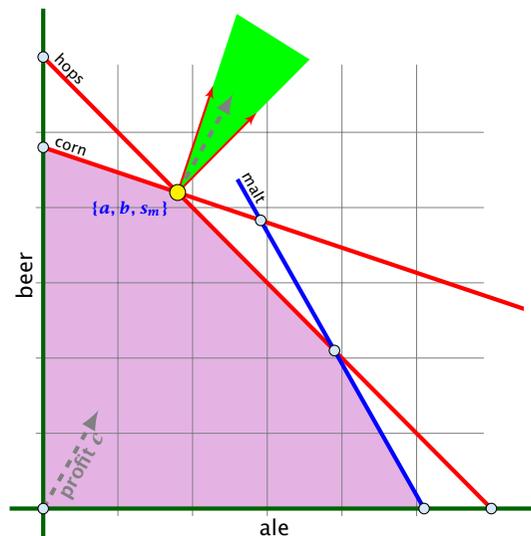


The profit vector $c$ lies in the cone generated by the normals for the hops and the corn constraint.

## Strong Duality

**Theorem 30 (Strong Duality)**

Let $P$ and $D$ be a primal dual pair of linear programs, and let $z^*$ and $w^*$ denote the optimal solution to $P$ and $D$, respectively. Then

$$z^* = w^*$$

## Strong Duality

### Theorem 31 (Strong Duality)

Let $P$ and $D$ be a primal dual pair of linear programs, and let $z^*$ and $w^*$ denote the optimal solution to $P$ and $D$, respectively. Then

$$z^* = w^*$$

### Lemma 32 (Weierstrass)

Let $X$ be a compact set and let $f(x)$ be a continuous function on $X$. Then $\min\{f(x) : x \in X\}$ exists.

### Lemma 33 (Projection Lemma)

Let $X \subseteq \mathbb{R}^m$ be a non-empty convex set, and let $y \notin X$. Then there exist $x^* \in X$ with minimum distance from $y$. Moreover for all $x \in X$ we have $(y - x^*)^T (x - x^*) \leq 0$.
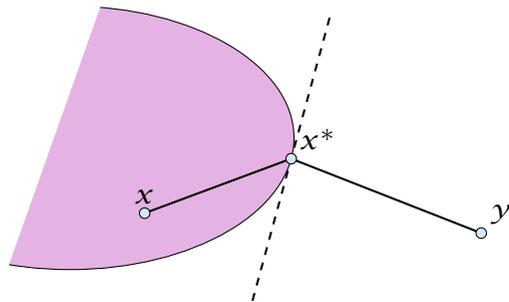
## Proof of the Projection Lemma

- Define $f(x) = \|y - x\|$.
- We want to apply Weierstrass but $X$ may not be bounded.
- $X \neq \emptyset$. Hence, there exists $x' \in X$.
- Define $X' = \{x \in X \mid \|y - x\| \leq \|y - x'\|\}$. This set is closed and bounded.
- Applying Weierstrass gives the existence.
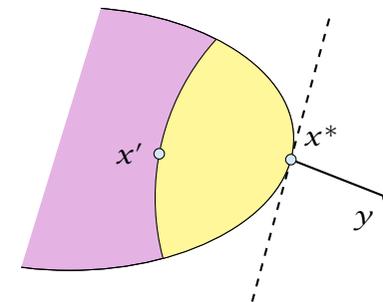
## Proof of the Projection Lemma (continued)

$x^*$ is minimum. Hence $\|y - x^*\|^2 \leq \|y - x\|^2$ for all $x \in X$.

By convexity: $x \in X$ then $x^* + \epsilon(x - x^*) \in X$ for all $0 \leq \epsilon \leq 1$.

$$\|y - x^*\|^2 \leq \|y - x^* - \epsilon(x - x^*)\|^2$$
$$= \|y - x^*\|^2 + \epsilon^2\|x - x^*\|^2 - 2\epsilon(y - x^*)^T(x - x^*)$$

Hence, $(y - x^*)^T(x - x^*) \leq \frac{1}{2}\epsilon\|x - x^*\|^2$.

Letting $\epsilon \to 0$ gives the result.

---

**Theorem 34 (Separating Hyperplane)**

*Let $X \subseteq \mathbb{R}^m$ be a non-empty closed convex set, and let $y \notin X$. Then there exists a separating hyperplane $\{x \in \mathbb{R} : a^T x = \alpha\}$ where $a \in \mathbb{R}^m$, $\alpha \in \mathbb{R}$ that separates $y$ from $X$. ($a^T y < \alpha$; $a^T x \geq \alpha$ for all $x \in X$)*

---

## Proof of the Hyperplane Lemma

- Let $x^* \in X$ be closest point to $y$ in $X$.
- By previous lemma $(y - x^*)^T(x - x^*) \leq 0$ for all $x \in X$.
- Choose $a = (x^* - y)$ and $\alpha = a^T x^*$.
- For $x \in X : a^T(x - x^*) \geq 0$, and, hence, $a^T x \geq \alpha$.
- Also, $a^T y = a^T(x^* - a) = \alpha - \|a\|^2 < \alpha$



$$H = \{x \mid a^T x = \alpha\}$$

---
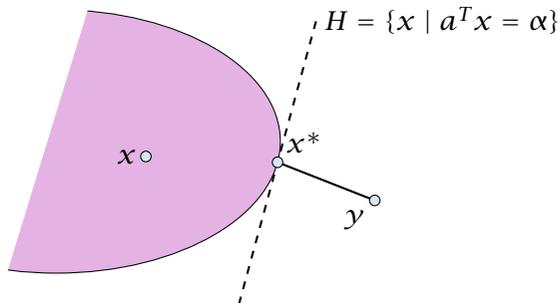
**Lemma 35 (Farkas Lemma)**

*Let $A$ be an $m \times n$ matrix, $b \in \mathbb{R}^m$. Then exactly one of the following statements holds.*

1. $\exists x \in \mathbb{R}^n$ with $Ax = b$, $x \geq 0$
2. $\exists y \in \mathbb{R}^m$ with $A^T y \geq 0$, $b^T y < 0$

Assume $\hat{x}$ satisfies 1. and $\hat{y}$ satisfies 2. Then

$$0 > y^T b = y^T A x \geq 0$$

Hence, at most one of the statements can hold.

## Proof of Farkas Lemma

Now, assume that 1. does not hold.

Consider $S = \{Ax : x \geq 0\}$ so that $S$ closed, convex, $b \notin S$.

We want to show that there is $y$ with $A^T y \geq 0$, $b^T y < 0$.

Let $y$ be a hyperplane that separates $b$ from $S$. Hence, $y^T b < \alpha$ and $y^T s \geq \alpha$ for all $s \in S$.

$0 \in S \Rightarrow \alpha \leq 0 \Rightarrow y^T b < 0$

$y^T Ax \geq \alpha$ for all $x \geq 0$. Hence, $y^T A \geq 0$ as we can choose $x$ arbitrarily large.

---

### Lemma 36 (Farkas Lemma; different version)

*Let $A$ be an $m \times n$ matrix, $b \in \mathbb{R}^m$. Then exactly one of the following statements holds.*

1. $\exists x \in \mathbb{R}^n$ with $Ax \leq b$, $x \geq 0$
2. $\exists y \in \mathbb{R}^m$ with $A^T y \geq 0$, $b^T y < 0$, $y \geq 0$

**Rewrite the conditions:**

1. $\exists x \in \mathbb{R}^n$ with $\begin{bmatrix} A\ I \end{bmatrix} \cdot \begin{bmatrix} x \\ s \end{bmatrix} = b$, $x \geq 0$, $s \geq 0$

2. $\exists y \in \mathbb{R}^m$ with $\begin{bmatrix} A^T \\ I \end{bmatrix} y \geq 0$, $b^T y < 0$

---

## Proof of Strong Duality

$P$: $z = \max\{c^T x \mid Ax \leq b, x \geq 0\}$

$D$: $w = \min\{b^T y \mid A^T y \geq c, y \geq 0\}$

### Theorem 37 (Strong Duality)

*Let $P$ and $D$ be a primal dual pair of linear programs, and let $z$ and $w$ denote the optimal solution to $P$ and $D$, respectively (i.e., $P$ and $D$ are non-empty). Then*

$$z = w \ .$$

---

## Proof of Strong Duality

$z \leq w$: follows from weak duality

$z \geq w$:
We show $z < \alpha$ implies $w < \alpha$.

| $\exists x \in \mathbb{R}^n$ | | | |
|---|---|---|---|
| s.t. | $Ax$ | $\leq$ | $b$ |
| | $-c^T x$ | $\leq$ | $-\alpha$ |
| | $x$ | $\geq$ | $0$ |

| $\exists y \in \mathbb{R}^m; v \in \mathbb{R}$ | | | |
|---|---|---|---|
| s.t. | $A^T y - cv$ | $\geq$ | $0$ |
| | $b^T y - \alpha v$ | $<$ | $0$ |
| | $y, v$ | $\geq$ | $0$ |

From the definition of $\alpha$ we know that the first system is infeasible; hence the second must be feasible.

## Proof of Strong Duality

$$\exists y \in \mathbb{R}^m; v \in \mathbb{R}$$
$$\text{s.t.} \quad A^T y - v \;\geq\; 0$$
$$b^T y - \alpha v \;<\; 0$$
$$y, v \;\geq\; 0$$

If the solution $y, v$ has $v = 0$ we have that

$$\exists y \in \mathbb{R}^m$$
$$\text{s.t.} \quad A^T y \;\geq\; 0$$
$$b^T y \;<\; 0$$
$$y \;\geq\; 0$$

is feasible. By Farkas lemma this gives that LP $P$ is infeasible. Contradiction to the assumption of the lemma.

## Proof of Strong Duality

Hence, there exists a solution $y, v$ with $v > 0$.

We can rescale this solution (scaling both $y$ and $v$) s.t. $v = 1$.

Then $y$ is feasible for the dual but $b^T y < \alpha$. This means that $w < \alpha$.

## Fundamental Questions

### Definition 38 (Linear Programming Problem (LP))

Let $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, $\alpha \in \mathbb{Q}$. Does there exist $x \in \mathbb{Q}^n$ s.t. $Ax = b$, $x \geq 0$, $c^T x \geq \alpha$?

**Questions**:
- Is LP in NP?
- Is LP in co-NP? yes!
- Is LP in P?

**Proof**:
- Given a primal maximization problem $P$ and a parameter $\alpha$. Suppose that $\alpha > \text{opt}(P)$.
- We can prove this by providing an optimal basis for the dual.
- A verifier can check that the associated dual solution fulfills all dual constraints and that it has dual cost $< \alpha$.

## Complementary Slackness

### Lemma 39

*Assume a linear program $P = \max\{c^T x \mid Ax \leq b; x \geq 0\}$ has solution $x^*$ and its dual $D = \min\{b^T y \mid A^T y \geq c; y \geq 0\}$ has solution $y^*$.*

1. *If $x_j^* > 0$ then the $j$-th constraint in $D$ is tight.*
2. *If the $j$-th constraint in $D$ is not tight than $x_j^* = 0$.*
3. *If $y_i^* > 0$ then the $i$-th constraint in $P$ is tight.*
4. *If the $i$-th constraint in $P$ is not tight than $y_i^* = 0$.*

If we say that a variable $x_j^*$ ($y_i^*$) has slack if $x_j^* > 0$ ($y_i^* > 0$), (i.e., the corresponding variable restriction is not tight) and a contraint has slack if it is not tight, then the above says that for a primal-dual solution pair it is not possible that a constraint **and** its corresponding (dual) variable has slack.

# Proof: Complementary Slackness

Analogous to the proof of weak duality we obtain

$$c^T x^* \leq y^{*T} A x^* \leq b^T y^*$$

Because of strong duality we then get

$$c^T x^* = y^{*T} A x^* = b^T y^*$$

This gives e.g.

$$\sum_j (y^T A - c^T)_j x_j^* = 0$$

From the constraint of the dual it follows that $y^T A \geq c^T$. Hence the left hand side is a sum over the product of non-negative numbers. Hence, if e.g. $(y^T A - c^T)_j > 0$ (the $j$-th constraint in the dual is not tight) then $x_j = 0$ (2.). The result for (1./3./4.) follows similarly.

# Interpretation of Dual Variables

- Brewer: find mix of ale and beer that maximizes profits

$$
\begin{aligned}
\max \quad & 13a \;+\; 23b \\
\text{s.t.} \quad & 5a \;+\; 15b \leq 480 \\
& 4a \;+\; 4b \leq 160 \\
& 35a \;+\; 20b \leq 1190 \\
& a, b \geq 0
\end{aligned}
$$

- Entrepeneur: buy resources from brewer at minimum cost
  $C, H, M$: unit price for corn, hops and malt.

$$
\begin{aligned}
\min \quad & 480C \;+\; 160H \;+\; 1190M \\
\text{s.t.} \quad & 5C \;+\; 4H \;+\; 35M \geq 13 \\
& 15C \;+\; 4H \;+\; 20M \geq 23 \\
& C, H, M \geq 0
\end{aligned}
$$

Note that brewer won't sell (at least not all) if e.g.
$5C + 4H + 35M < 13$ as then brewing ale would be advantageous.

# Interpretation of Dual Variables

**Marginal Price:**

- How much money is the brewer willing to pay for additional amount of Corn, Hops, or Malt?
- We are interested in the marginal price, i.e., what happens if we increase the amount of Corn, Hops, and Malt by $\varepsilon_C, \varepsilon_H$, and $\varepsilon_M$, respectively.

The profit increases to $\max\{c^T x \mid Ax \leq b + \varepsilon; x \geq 0\}$. Because of strong duality this is equal to

$$
\begin{aligned}
\min \quad & (b^T + \epsilon^T) y \\
\text{s.t.} \quad & A^T y \;\geq\; c \\
& y \;\geq\; 0
\end{aligned}
$$

# Interpretation of Dual Variables

If $\epsilon$ is "small" enough then the optimum dual solution $y^*$ might not change. Therefore the profit increases by $\sum_i \varepsilon_i y_i^*$.

Therefore we can interpret the dual variables as marginal prices.

Note that with this interpretation, complementary slackness becomes obvious.

- If the brewer has slack of some resource (e.g. corn) then he is not willing to pay anything for it (corresponding dual variable is zero).
- If the dual variable for some resource is non-zero, then an increase of this resource increases the profit of the brewer. Hence, it makes no sense to have left-overs of this resource. Therefore its slack must be zero.

## Example



max  $13a + 23b$

s.t.  $5a + 15b + s_c \qquad\qquad = 480$

$\qquad 4a + 4b \qquad + s_h \qquad = 160$

$\qquad 35a + 20b \qquad\qquad + s_m = 1190$

$\qquad a \ , \quad b \ , s_c \ , \ s_h \ , \ s_m \geq 0$
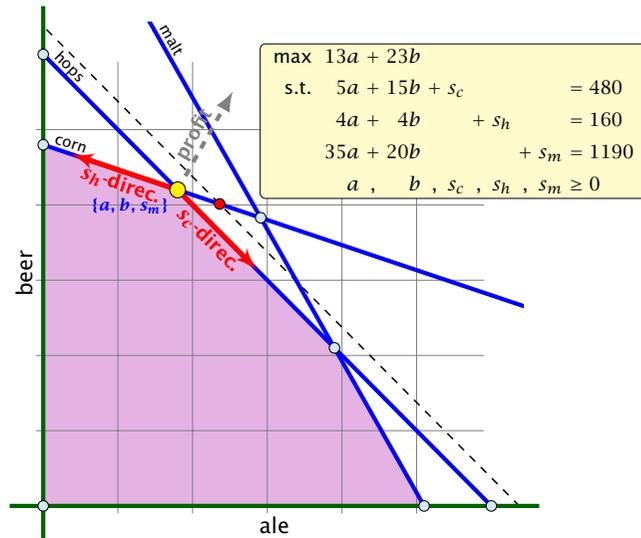
The change in profit when increasing hops by one unit is

$= \underbrace{c_B^T A_B^{-1}}_{y^*} e_h.$

---

Of course, the previous argument about the increase in the primal objective only holds for the non-degenerate case.

If the optimum basis is degenerate then increasing the supply of one resource may not allow the objective value to increase.

---

## Flows

### Definition 40
An $(s,t)$-flow in a (complete) directed graph $G = (V, V \times V, c)$ is a function $f : V \times V \mapsto \mathbb{R}_0^+$ that satisfies

1. For each edge $(x, y)$

$$0 \leq f_{xy} \leq c_{xy} \ .$$

(capacity constraints)

2. For each $v \in V \setminus \{s, t\}$

$$\sum_x f_{vx} = \sum_x f_{xv} \ .$$

(flow conservation constraints)

---

## Flows

### Definition 41
The value of an $(s,t)$-flow $f$ is defined as

$$\text{val}(f) = \sum_x f_{sx} - \sum_x f_{xs} \ .$$

**Maximum Flow Problem:**
Find an $(s,t)$-flow with maximum value.

## LP-Formulation of Maxflow

$$
\begin{aligned}
\max \quad & \sum_z f_{sz} - \sum_z f_{zs} \\
\text{s.t.} \quad \forall (z,w) \in V \times V \quad & f_{zw} \le c_{zw} \quad \ell_{zw} \\
\forall w \ne s,t \quad & \sum_z f_{zw} - \sum_z f_{wz} = 0 \quad p_w \\
& f_{zw} \ge 0
\end{aligned}
$$

$$
\begin{array}{llrcl}
\min & & \multicolumn{3}{c}{\sum_{(xy)} c_{xy}\ell_{xy}} \\
\text{s.t.} & f_{xy}\,(x,y \ne s,t): & 1\ell_{xy} - 1p_x + 1p_y & \ge & 0 \\
& f_{sy}\,(y \ne s,t): & 1\ell_{sy}\ \ +1p_y & \ge & 1 \\
& f_{xs}\,(x \ne s,t): & 1\ell_{xs} - 1p_x & \ge & -1 \\
& f_{ty}\,(y \ne s,t): & 1\ell_{ty}\ \ +1p_y & \ge & 0 \\
& f_{xt}\,(x \ne s,t): & 1\ell_{xt} - 1p_x & \ge & 0 \\
& f_{st}: & 1\ell_{st} & \ge & 1 \\
& f_{ts}: & 1\ell_{ts} & \ge & -1 \\
& & \ell_{xy} & \ge & 0
\end{array}
$$

## LP-Formulation of Maxflow

$$
\begin{array}{llrcl}
\min & & \multicolumn{3}{c}{\sum_{(xy)} c_{xy}\ell_{xy}} \\
\text{s.t.} & f_{xy}\,(x,y \ne s,t): & 1\ell_{xy} - 1p_x + 1p_y & \ge & 0 \\
& f_{sy}\,(y \ne s,t): & 1\ell_{sy} - \ \ 1 + 1p_y & \ge & 0 \\
& f_{xs}\,(x \ne s,t): & 1\ell_{xs} - 1p_x + \ \ 1 & \ge & 0 \\
& f_{ty}\,(y \ne s,t): & 1\ell_{ty} - \ \ 0 + 1p_y & \ge & 0 \\
& f_{xt}\,(x \ne s,t): & 1\ell_{xt} - 1p_x + \ \ 0 & \ge & 0 \\
& f_{st}: & 1\ell_{st} - \ \ 1 + \ \ 0 & \ge & 0 \\
& f_{ts}: & 1\ell_{ts} - \ \ 0 + \ \ 1 & \ge & 0 \\
& & \ell_{xy} & \ge & 0
\end{array}
$$

## LP-Formulation of Maxflow

$$
\begin{array}{llrcl}
\min & & \multicolumn{3}{c}{\sum_{(xy)} c_{xy}\ell_{xy}} \\
\text{s.t.} & f_{xy}\,(x,y \ne s,t): & 1\ell_{xy} - 1p_x + 1p_y & \ge & 0 \\
& f_{sy}\,(y \ne s,t): & 1\ell_{sy} - \ \ p_s + 1p_y & \ge & 0 \\
& f_{xs}\,(x \ne s,t): & 1\ell_{xs} - 1p_x + \ \ p_s & \ge & 0 \\
& f_{ty}\,(y \ne s,t): & 1\ell_{ty} - \ \ p_t + 1p_y & \ge & 0 \\
& f_{xt}\,(x \ne s,t): & 1\ell_{xt} - 1p_x + \ \ p_t & \ge & 0 \\
& f_{st}: & 1\ell_{st} - \ \ p_s + \ \ p_t & \ge & 0 \\
& f_{ts}: & 1\ell_{ts} - \ \ p_t + \ \ p_s & \ge & 0 \\
& & \ell_{xy} & \ge & 0
\end{array}
$$

with $p_t = 0$ and $p_s = 1$.

## LP-Formulation of Maxflow

$$
\begin{array}{llrcl}
\min & & \multicolumn{3}{c}{\sum_{(xy)} c_{xy}\ell_{xy}} \\
\text{s.t.} & f_{xy}: & 1\ell_{xy} - 1p_x + 1p_y & \ge & 0 \\
& & \ell_{xy} & \ge & 0 \\
& & p_s & = & 1 \\
& & p_t & = & 0
\end{array}
$$

We can interpret the $\ell_{xy}$ value as assigning a length to every edge.

The value $p_x$ for a variable, then can be seen as the distance of $x$ to $t$ (where the distance from $s$ to $t$ is required to be 1 since $p_s = 1$).

The constraint $p_x \le \ell_{xy} + p_y$ then simply follows from triangle inequality ($d(x,t) \le d(x,y) + d(y,t) \Rightarrow d(x,t) \le \ell_{xy} + d(y,t)$).

One can show that there is an optimum LP-solution for the dual problem that gives an integral assignment of variables.

This means $p_x = 1$ or $p_x = 0$ for our case. This gives rise to a cut in the graph with vertices having value $1$ on one side and the other vertices on the other side. The objective function then evaluates the capacity of this cut.

This shows that the Maxflow/Mincut theorem follows from linear programming duality.

## Flows

### Definition 42
An $(s,t)$-flow in a (complete) directed graph $G = (V, V \times V, c)$ is a function $f : V \times V \mapsto \mathbb{R}_0^+$ that satisfies

1. For each edge $(x, y)$

$$0 \le f_{xy} \le c_{xy} \ .$$

   (capacity constraints)
2. For each $v \in V \setminus \{s, t\}$

$$\sum_x f_{vx} = \sum_x f_{xv} \ .$$

   (flow conservation constraints)

## Flows

### Definition 43
The value of an $(s,t)$-flow $f$ is defined as

$$\mathrm{val}(f) = \sum_x f_{sx} - \sum_x f_{xs} \ .$$

**Maximum Flow Problem:**
Find an $(s,t)$-flow with maximum value.

## LP-Formulation of Maxflow

$$
\begin{array}{lllcll}
\max & & \sum_z f_{sz} - \sum_z f_{zs} & & & \\
\text{s.t.} & \forall (z, w) \in V \times V & f_{zw} & \le & c_{zw} & \ell_{zw} \\
& \forall w \ne s, t & \sum_z f_{zw} - \sum_z f_{wz} & = & 0 & p_w \\
& & f_{zw} & \ge & 0 &
\end{array}
$$

$$
\begin{array}{llllcl}
\min & & & \sum_{(xy)} c_{xy}\ell_{xy} & & \\
\text{s.t.} & f_{xy}\ (x, y \ne s, t): & 1\ell_{xy} - 1p_x + 1p_y & & \ge & 0 \\
& f_{sy}\ (y \ne s, t): & 1\ell_{sy} + 1p_y & & \ge & 1 \\
& f_{xs}\ (x \ne s, t): & 1\ell_{xs} - 1p_x & & \ge & -1 \\
& f_{ty}\ (y \ne s, t): & 1\ell_{ty} + 1p_y & & \ge & 0 \\
& f_{xt}\ (x \ne s, t): & 1\ell_{xt} - 1p_x & & \ge & 0 \\
& f_{st}: & 1\ell_{st} & & \ge & 1 \\
& f_{ts}: & 1\ell_{ts} & & \ge & -1 \\
& & \ell_{xy} & & \ge & 0
\end{array}
$$

## LP-Formulation of Maxflow

$$
\begin{array}{lllrcl}
\min & & \sum_{(xy)} c_{xy}\ell_{xy} & & & \\
\text{s.t.} & f_{xy}\ (x,y \neq s,t): & 1\ell_{xy}-1p_x+1p_y & \geq & 0 \\
& f_{sy}\ (y \neq s,t): & 1\ell_{sy}- & 1+1p_y & \geq & 0 \\
& f_{xs}\ (x \neq s,t): & 1\ell_{xs}-1p_x+ & 1 & \geq & 0 \\
& f_{ty}\ (y \neq s,t): & 1\ell_{ty}- & 0+1p_y & \geq & 0 \\
& f_{xt}\ (x \neq s,t): & 1\ell_{xt}-1p_x+ & 0 & \geq & 0 \\
& f_{st}: & 1\ell_{st}- & 1+ & 0 & \geq & 0 \\
& f_{ts}: & 1\ell_{ts}- & 0+ & 1 & \geq & 0 \\
& & & \ell_{xy} & \geq & 0
\end{array}
$$

## LP-Formulation of Maxflow

$$
\begin{array}{lllrcl}
\min & & \sum_{(xy)} c_{xy}\ell_{xy} & & & \\
\text{s.t.} & f_{xy}\ (x,y \neq s,t): & 1\ell_{xy}-1p_x+1p_y & \geq & 0 \\
& f_{sy}\ (y \neq s,t): & 1\ell_{sy}- & p_s+1p_y & \geq & 0 \\
& f_{xs}\ (x \neq s,t): & 1\ell_{xs}-1p_x+ & p_s & \geq & 0 \\
& f_{ty}\ (y \neq s,t): & 1\ell_{ty}- & p_t+1p_y & \geq & 0 \\
& f_{xt}\ (x \neq s,t): & 1\ell_{xt}-1p_x+ & p_t & \geq & 0 \\
& f_{st}: & 1\ell_{st}- & p_s+ & p_t & \geq & 0 \\
& f_{ts}: & 1\ell_{ts}- & p_t+ & p_s & \geq & 0 \\
& & & \ell_{xy} & \geq & 0
\end{array}
$$

with $p_t = 0$ and $p_s = 1$.

## LP-Formulation of Maxflow

$$
\begin{array}{llrcl}
\min & & \sum_{(xy)} c_{xy}\ell_{xy} & & \\
\text{s.t.} & f_{xy}: & 1\ell_{xy}-1p_x+1p_y & \geq & 0 \\
& & \ell_{xy} & \geq & 0 \\
& & p_s & = & 1 \\
& & p_t & = & 0
\end{array}
$$

We can interpret the $\ell_{xy}$ value as assigning a length to every edge.

The value $p_x$ for a variable, then can be seen as the distance of $x$ to $t$ (where the distance from $s$ to $t$ is required to be 1 since $p_s = 1$).

The constraint $p_x \leq \ell_{xy} + p_y$ then simply follows from triangle inequality ($d(x,t) \leq d(x,y) + d(y,t) \Rightarrow d(x,t) \leq \ell_{xy} + d(y,t)$).

One can show that there is an optimum LP-solution for the dual problem that gives an integral assignment of variables.

This means $p_x = 1$ or $p_x = 0$ for our case. This gives rise to a cut in the graph with vertices having value $1$ on one side and the other vertices on the other side. The objective function then evaluates the capacity of this cut.

This shows that the Maxflow/Mincut theorem follows from linear programming duality.

## Degeneracy Revisited

If a basis variable is $0$ in the basic feasible solution then we may not make progress during an iteration of simplex.

**Idea:**

Change $\mathrm{LP} := \max\{c^T x, Ax = b; x \geq 0\}$ into
$\mathrm{LP}' := \max\{c^T x, Ax = b', x \geq 0\}$ such that

  **I.** LP is feasible

  **II.** If a set $B$ of basis variables corresponds to an infeasible basis (i.e. $A_B^{-1} b \not\geq 0$) then $B$ corresponds to an infeasible basis in $\mathrm{LP}'$ (note that columns in $A_B$ are linearly independent).

  **III.** LP has no degenerate basic solutions

## Degenerate Example
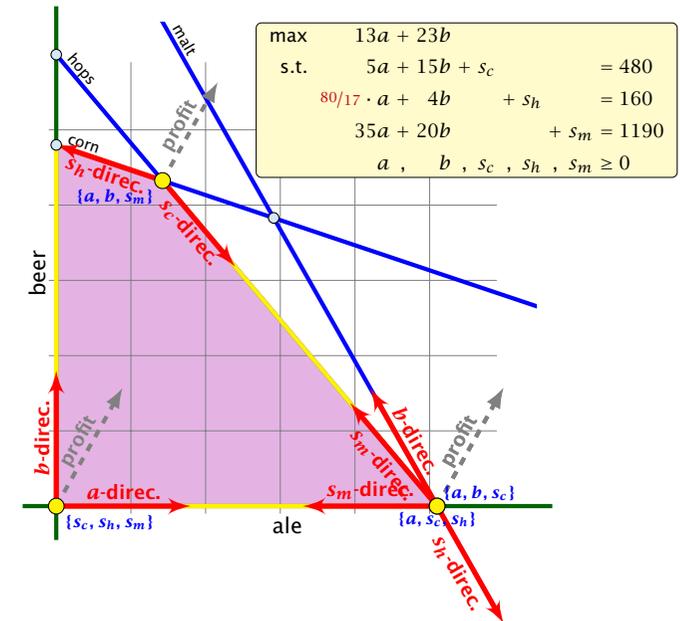


$$\begin{aligned}
\max \quad & 13a + 23b \\
\text{s.t.} \quad & 5a + 15b + s_c && = 480 \\
& {}^{80}\!/_{17} \cdot a + 4b && + s_h && = 160 \\
& 35a + 20b && + s_m = 1190 \\
& a, \quad b, \quad s_c, \quad s_h, \quad s_m \geq 0
\end{aligned}$$

## Degeneracy Revisited

If a basis variable is $0$ in the basic feasible solution then we may not make progress during an iteration of simplex.

**Idea:**

Given feasible $\mathrm{LP} := \max\{c^T x, Ax = b; x \geq 0\}$. Change it into $\mathrm{LP}' := \max\{c^T x, Ax = b', x \geq 0\}$ such that

  **I.** $\mathrm{LP}'$ is feasible

  **II.** If a set $B$ of basis variables corresponds to an infeasible basis (i.e. $A_B^{-1} b \not\geq 0$) then $B$ corresponds to an infeasible basis in $\mathrm{LP}'$ (note that columns in $A_B$ are linearly independent).

  **III.** $\mathrm{LP}'$ has no degenerate basic solutions

## Perturbation

Let $B$ be index set of some basis with basic solution

$$x_B^* = A_B^{-1} b \geq 0, x_N^* = 0 \quad \text{(i.e. } B \text{ is feasible)}$$

Fix

$$b' := b + A_B \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix} \text{ for } \varepsilon > 0 \ .$$

This is the perturbation that we are using.

# Property I

The new LP is feasible because the set $B$ of basis variables provides a feasible basis:

$$A_B^{-1}\left(b + A_B \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}\right) = x_B^* + \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix} \geq 0 \ .$$

# Property II

Let $\tilde{B}$ be a non-feasible basis. This means $(A_{\tilde{B}}^{-1} b)_i < 0$ for some row $i$.

Then for small enough $\epsilon > 0$

$$\left(A_{\tilde{B}}^{-1}\left(b + A_B \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}\right)\right)_i = (A_{\tilde{B}}^{-1} b)_i + \left(A_{\tilde{B}}^{-1} A_B \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}\right)_i < 0$$

Hence, $\tilde{B}$ is not feasible.

# Property III

Let $\tilde{B}$ be a basis. It has an associated solution

$$x_{\tilde{B}}^* = A_{\tilde{B}}^{-1} b + A_{\tilde{B}}^{-1} A_B \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}$$

in the perturbed instance.

We can view each component of the vector as a polynom with variable $\varepsilon$ of degree at most $m$.

$A_{\tilde{B}}^{-1} A_B$ has rank $m$. Therefore no polynom is $0$.

A polynom of degree at most $m$ has at most $m$ roots (Nullstellen).

Hence, $\epsilon > 0$ small enough gives that no component of the above vector is $0$. Hence, no degeneracies.

Since, there are no degeneracies Simplex will terminate when run on $\text{LP}'$.

▶ If it terminates because the reduced cost vector fulfills

$$\tilde{c} = (c^T - c_B^T A_B^{-1} A) \leq 0$$

then we have found an optimal basis. Note that this basis is also optimal for LP, as the above constraint does not depend on $b$.

▶ If it terminates because it finds a variable $x_j$ with $\tilde{c}_j > 0$ for which the $j$-th basis direction $d$, fulfills $d \geq 0$ we know that $\text{LP}'$ is unbounded. The basis direction does not depend on $b$. Hence, we also know that LP is unbounded.

## Lexicographic Pivoting

Doing calculations with perturbed instances may be costly. Also the right choice of $\varepsilon$ is difficult.

**Idea:**
Simulate behaviour of $\mathrm{LP}'$ without explicitly doing a perturbation.

## Lexicographic Pivoting

We choose the entering variable arbitrarily as before ($\tilde{c}_e > 0$, of course).

If we do not have a choice for the leaving variable then $\mathrm{LP}'$ and $\mathrm{LP}$ do the same (i.e., choose the same variable).

Otherwise we have to be careful.

## Lexicographic Pivoting

In the following we assume that $b \geq 0$. This can be obtained by replacing the initial system $(A_B \mid b)$ by $(A_B^{-1}A \mid A_B^{-1}b)$ where $B$ is the index set of a feasible basis (found e.g. by the first phase of the Two-phase algorithm).

Then the perturbed instance is

$$b' = b + \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}$$

## Matrix View

Let our linear program be

$$
\begin{array}{rcrcl}
c_B^T x_B & + & c_N^T x_N & = & Z \\
A_B x_B & + & A_N x_N & = & b \\
x_B & , & x_N & \geq & 0
\end{array}
$$

The simplex tableaux for basis $B$ is

$$
\begin{array}{rcrcl}
& & (c_N^T - c_B^T A_B^{-1} A_N) x_N & = & Z - c_B^T A_B^{-1} b \\
I x_B & + & A_B^{-1} A_N x_N & = & A_B^{-1} b \\
x_B & , & x_N & \geq & 0
\end{array}
$$

The BFS is given by $x_N = 0, x_B = A_B^{-1}b$.

If $(c_N^T - c_B^T A_B^{-1} A_N) \leq 0$ we know that we have an optimum solution.

## Lexicographic Pivoting

LP chooses an arbitrary leaving variable that has $\hat{A}_{\ell e} > 0$ and minimizes

$$\theta_\ell = \frac{\hat{b}_\ell}{\hat{A}_{\ell e}} = \frac{(A_B^{-1}b)_\ell}{(A_B^{-1}A_{*e})_\ell} \;.$$

$\ell$ is the index of a leaving variable within $B$. This means if e.g. $B = \{1, 3, 7, 14\}$ and leaving variable is $3$ then $\ell = 2$.

## Lexicographic Pivoting

**Definition 44**
$u \leq_{\text{lex}} v$ if and only if the first component in which $u$ and $v$ differ fulfills $u_i \leq v_i$.

## Lexicographic Pivoting

LP' chooses an index that minimizes

$$\theta_\ell = \frac{\left(A_B^{-1}\left(b + \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}\right)\right)_\ell}{(A_B^{-1}A_{*e})_\ell} = \frac{\left(A_B^{-1}(b \mid I)\begin{pmatrix} 1 \\ \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}\right)_\ell}{(A_B^{-1}A_{*e})_\ell}$$

$$= \frac{\ell\text{-th row of } A_B^{-1}(b \mid I)}{(A_B^{-1}A_{*e})_\ell}\begin{pmatrix} 1 \\ \varepsilon \\ \vdots \\ \varepsilon^m \end{pmatrix}$$

## Lexicographic Pivoting

This means you can choose the variable/row $\ell$ for which the vector

$$\frac{\ell\text{-th row of } A_B^{-1}(b \mid I)}{(A_B^{-1}A_{*e})_\ell}$$

is lexicographically minimal.

Of course only including rows with $(A_B^{-1}A_{*e})_\ell > 0$.

This technique guarantees that your pivoting is the same as in the perturbed case. This guarantees that cycling does not occur.

## Number of Simplex Iterations

Each iteration of Simplex can be implemented in polynomial time.

If we use lexicographic pivoting we know that Simplex requires at most $\binom{n}{m}$ iterations, because it will not visit a basis twice.

The input size is $L \cdot n \cdot m$, where $n$ is the number of variables, $m$ is the number of constraints, and $L$ is the length of the binary representation of the largest coefficient in the matrix $A$.

If we really require $\binom{n}{m}$ iterations then Simplex is not a polynomial time algorithm.

**Can we obtain a better analysis?**

## Number of Simplex Iterations

**Observation**
Simplex visits every feasible basis at most once.

However, also the number of feasible bases can be very large.

## Example

$$\max c^T x$$
$$\text{s.t. } 0 \le x_1 \le 1$$
$$0 \le x_2 \le 1$$
$$\vdots$$
$$0 \le x_n \le 1$$
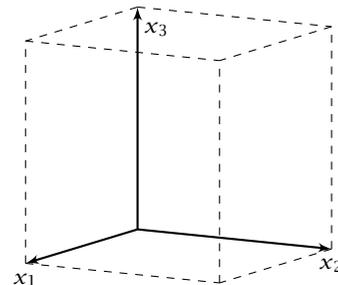


$2n$ constraint on $n$ variables define an $n$-dimensional hypercube as feasible region.

The feasible region has $2^n$ vertices.

## Example

$$\max c^T x$$
$$\text{s.t. } 0 \le x_1 \le 1$$
$$0 \le x_2 \le 1$$
$$\vdots$$
$$0 \le x_n \le 1$$
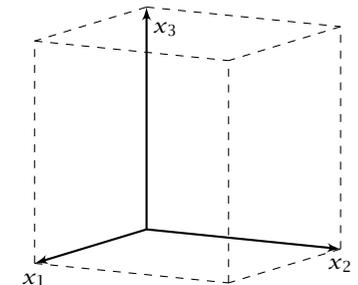


However, Simplex may still run quickly as it usually does not visit all feasible bases.

In the following we give an example of a feasible region for which there is a bad Pivoting Rule.

## Pivoting Rule

A Pivoting Rule defines how to choose the entering and leaving variable for an iteration of Simplex.

In the non-degenerate case after choosing the entering variable the leaving variable is unique.

## Klee Minty Cube

$$\begin{aligned}
\max \ & x_n \\
\text{s.t.} \quad & 0 \le x_1 \le 1 \\
& \epsilon x_1 \le x_2 \le 1 - \epsilon x_1 \\
& \epsilon x_2 \le x_3 \le 1 - \epsilon x_2 \\
& \qquad\qquad \vdots \\
& \epsilon x_{n-1} \le x_n \le 1 - \epsilon x_{n-1} \\
& x_i \ge 0
\end{aligned}$$
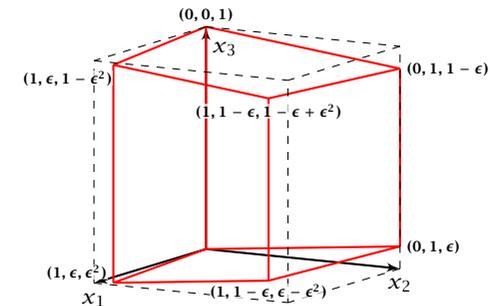
## Observations

▶ We have $2n$ constraints, and $3n$ variables (after adding slack variables to every constraint).

▶ Every basis is defined by $2n$ variables, and $n$ non-basic variables.

▶ There exist degenerate vertices.

▶ The degeneracies come from the non-negativity constraints, which are superfluous.

▶ In the following all variables $x_i$ stay in the basis at all times.

▶ Then, we can uniquely specify a basis by choosing for each variable whether it should be equal to its lower bound, or equal to its upper bound (the slack variable corresponding to the non-tight constraint is part of the basis).

▶ We can also simply identify each basis/vertex with the corresponding hypercube vertex obtained by letting $\epsilon \to 0$.

## Analysis

▶ In the following we specify a sequence of bases (identified by the corresponding hypercube node) along which the objective function strictly increases.

▶ The basis $(0, \dots, 0, 1)$ is the unique optimal basis.

▶ Our sequence $S_n$ starts at $(0, \dots, 0)$ ends with $(0, \dots, 0, 1)$ and visits every node of the hypercube.

▶ An unfortunate Pivoting Rule may choose this sequence, and, hence, require an exponential number of iterations.

## Klee Minty Cube

max $x_n$
s.t.     $0 \le x_1 \le 1$
         $\epsilon x_1 \le x_2 \le 1 - \epsilon x_1$
         $\epsilon x_2 \le x_3 \le 1 - \epsilon x_2$



---
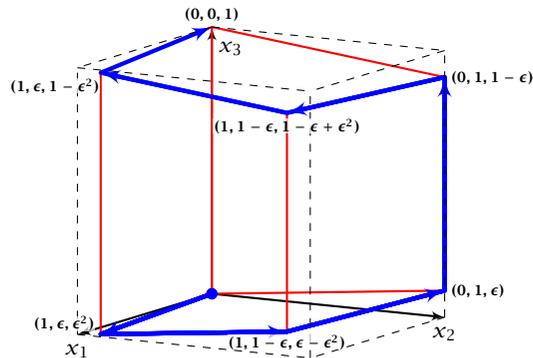
## Analysis

The sequence $S_n$ that visits every node of the hypercube is defined recursively



The non-recursive case is $S_1 = 0 \to 1$

---

## Analysis

**Lemma 45**

*The objective value $x_n$ is increasing along path $S_n$.*

**Proof by induction:**

$n = 1$: obvious, since $S_1 = 0 \to 1$, and $1 > 0$.

$n - 1 \to n$

- For the first part the value of $x_n = \epsilon x_{n-1}$.
- By induction hypothesis $x_{n-1}$ is increasing along $S_{n-1}$, hence, also $x_n$.
- Going from $(0, \ldots, 0, 1, 0)$ to $(0, \ldots, 0, 1, 1)$ increases $x_n$ for small enough $\epsilon$.
- For the remaining path $S_{n-1}^{\text{rev}}$ we have $x_n = 1 - \epsilon x_{n-1}$.
- By induction hypothesis $x_{n-1}$ is increasing along $S_{n-1}$, hence $-\epsilon x_{n-1}$ is increasing along $S_{n-1}^{\text{rev}}$.

---

## Remarks about Simplex

**Observation**

The simplex algorithm takes at most $\binom{n}{m}$ iterations. Each iteration can be implemented in time $\mathcal{O}(mn)$.

In practise it usually takes a linear number of iterations.

# Remarks about Simplex

**Theorem**
For almost all known deterministic pivoting rules (rules for choosing entering and leaving variables) there exist lower bounds that require the algorithm to have exponential running time ($\Omega(2^{\Omega(n)})$) (e.g. Klee Minty 1972).

# Remarks about Simplex

**Theorem**
For some standard randomized pivoting rules there exist subexponential lower bounds ($\Omega(2^{\Omega(n^{\alpha})})$ for $\alpha > 0$) (Friedmann, Hansen, Zwick 2011).

# Remarks about Simplex

**Conjecture** (Hirsch 1957)
The edge-vertex graph of an $m$-facet polytope in $d$-dimensional Euclidean space has diameter no more than $m - d$.

The conjecture has been proven wrong in 2010.

But the question whether the diameter is perhaps of the form $\mathcal{O}(\mathrm{poly}(m, d))$ is open.

# 8 Seidels LP-algorithm

▶ Suppose we want to solve $\min\{c^T x \mid Ax \geq b; x \geq 0\}$, where $x \in \mathbb{R}^d$ and we have $m$ constraints.
▶ In the worst-case Simplex runs in time roughly $\mathcal{O}(m(m + d)\binom{m+d}{m})) \approx (m + d)^m$. (slightly better bounds on the running time exist, but will not be discussed here).
▶ If $d$ is much smaller than $m$ one can do a lot better.
▶ In the following we develop an algorithm with running time $\mathcal{O}(d! \cdot m)$, i.e., linear in $m$.

# 8 Seidels LP-algorithm

**Setting:**

▶ We assume an LP of the form

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

▶ We assume that the LP is bounded.

# Ensuring Conditions

Given a standard minimization LP

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

how can we obtain an LP of the required form?

▶ **Compute a lower bound on $c^T x$ for any basic feasible solution.**

# Computing a Lower Bound

Let $s$ denote the smallest common multiple of all denominators of entries in $A, b$.

Multiply entries in $A, b$ by $s$ to obtain integral entries. This does not change the feasible region.

Add slack variables to $A$; denote the resulting matrix with $\bar{A}$.

If $B$ is an optimal basis then $x_B$ with $\bar{A}_B x_B = \bar{b}$, gives an optimal assignment to the basis variables (non-basic variables are $0$).

# Theorem 46 (Cramers Rule)

*Let $M$ be a matrix with $\det(M) \neq 0$. Then the solution to the system $Mx = b$ is given by*

$$
x_j = \frac{\det(M_j)}{\det(M)} \, ,
$$

*where $M_j$ is the matrix obtained from $M$ by replacing the $j$-th column by the vector $b$.*

**Proof:**

- Define

$$X_j = \begin{pmatrix} | & & | & | & | & & | \\ e_1 & \cdots & e_{j-1} & x & e_{j+1} & \cdots & e_n \\ | & & | & | & | & & | \end{pmatrix}$$

  Note that expanding along the $j$-th column gives that $\det(X_j) = x_j$.

- Further, we have

$$MX_j = \begin{pmatrix} | & & | & | & | & & | \\ Me_1 & \cdots & Me_{j-1} & Mx & Me_{j+1} & \cdots & Me_n \\ | & & | & | & | & & | \end{pmatrix} = M_j$$

- Hence,

$$x_j = \det(X_j) = \frac{\det(M_j)}{\det(M)}$$

---

## Bounding the Determinant

Let $Z$ be the maximum absolute entry occuring in $\bar{A}, \bar{b}$ or $c$. Let $C$ denote the matrix obtained from $\bar{A}_B$ by replacing the $j$-th column with vector $\bar{b}$.

Observe that

$$\begin{aligned} |\det(C)| &= \left| \sum_{\pi \in S_m} \operatorname{sgn}(\pi) \prod_{1 \le i \le m} C_{i\pi(i)} \right| \\ &\le \sum_{\pi \in S_m} \prod_{1 \le i \le m} |C_{i\pi(i)}| \\ &\le m! \cdot Z^m . \end{aligned}$$

---

## Bounding the Determinant

Alternatively, Hadamards inequality gives

$$\begin{aligned} |\det(C)| &\le \prod_{i=1}^{m} \|C_{*i}\| \le \prod_{i=1}^{m} (\sqrt{m} Z) \\ &\le m^{m/2} Z^m . \end{aligned}$$

---

## Hadamards Inequality
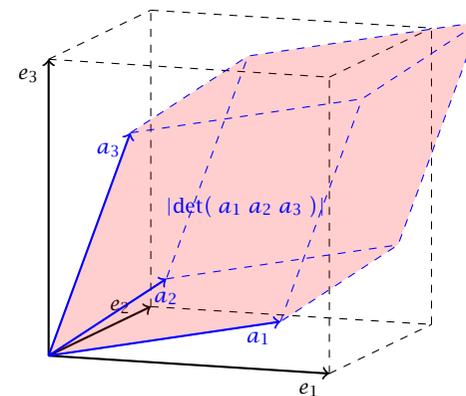


Hadamards inequality says that the volume of the red parallelepiped (Spat) is smaller than the volume in the black cube (if $\|e_1\| = \|a_1\|$, $\|e_2\| = \|a_2\|$, $\|e_3\| = \|a_3\|$).

## Ensuring Conditions

Given a standard minimization LP

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

how can we obtain an LP of the required form?

▶ **Compute a lower bound on $c^T x$ for any basic feasible solution.** Add the constraint $c^T x \geq -mZ(m! \cdot Z^m) - 1$. Note that this constraint is superfluous unless the LP is unbounded.

## Ensuring Conditions

Compute an optimum basis for the new LP.

▶ If the cost is $c^T x = -(mZ)(m! \cdot Z^m) - 1$ we know that the original LP is unbounded.

▶ Otw. we have an optimum basis.

In the following we use $\mathcal{H}$ to denote the set of all constraints apart from the constraint $c^T x \geq -mZ(m! \cdot Z^m) - 1$.

We give a routine SeidelLP$(\mathcal{H}, d)$ that is given a set $\mathcal{H}$ of explicit, non-degenerate constraints over $d$ variables, and minimizes $c^T x$ over all feasible points.

In addition it obeys the implicit constraint $c^T x \geq -(mZ)(m! \cdot Z^m) - 1$.

---

**Algorithm 1** SeidelLP$(\mathcal{H}, d)$

1: **if** $d = 1$ **then** solve 1-dimensional problem and return;
2: **if** $\mathcal{H} = \emptyset$ **then** return $x$ on implicit constraint hyperplane
3: choose random constraint $h \in \mathcal{H}$
4: $\hat{\mathcal{H}} \leftarrow \mathcal{H} \setminus \{h\}$
5: $\hat{x}^* \leftarrow$ SeidelLP$(\hat{\mathcal{H}}, d)$
6: **if** $\hat{x}^* =$ infeasible **then return** infeasible
7: **if** $\hat{x}^*$ fulfills $h$ **then return** $\hat{x}^*$
8: // optimal solution fulfills $h$ with equality, i.e., $a_h^T x = b_h$
9: solve $a_h^T x = b_h$ for some variable $x_\ell$;
10: eliminate $x_\ell$ in constraints from $\hat{\mathcal{H}}$ and in implicit constr.;
11: $\hat{x}^* \leftarrow$ SeidelLP$(\hat{\mathcal{H}}, d - 1)$
12: **if** $\hat{x}^* =$ infeasible **then**
13:      **return** infeasible
14: **else**
15:      add the value of $x_\ell$ to $\hat{x}^*$ and return the solution

# 8 Seidels LP-algorithm

- ▸ If $d = 1$ we can solve the 1-dimensional problem in time $\mathcal{O}(m)$.
- ▸ If $d > 1$ and $m = 0$ we take time $\mathcal{O}(d)$ to return $d$-dimensional vector $x$.
- ▸ The first recursive call takes time $T(m - 1, d)$ for the call plus $\mathcal{O}(d)$ for checking whether the solution fulfills $h$.
- ▸ If we are unlucky and $\hat{x}^*$ does not fulfill $h$ we need time $\mathcal{O}(d(m + 1)) = \mathcal{O}(dm)$ to eliminate $x_\ell$. Then we make a recursive call that takes time $T(m - 1, d - 1)$.
- ▸ The probability of being unlucky is at most $d/m$ as there are at most $d$ constraints whose removal will decrease the objective function

---

# 8 Seidels LP-algorithm

This gives the recurrence

$$T(m, d) = \begin{cases} \mathcal{O}(m) & \text{if } d = 1 \\ \mathcal{O}(d) & \text{if } d > 1 \text{ and } m = 0 \\ \mathcal{O}(d) + T(m - 1, d) + \\ \frac{d}{m}(\mathcal{O}(dm) + T(m - 1, d - 1)) & \text{otw.} \end{cases}$$

Note that $T(m, d)$ denotes the expected running time.

---

# 8 Seidels LP-algorithm

Let $C$ be the largest constant in the $\mathcal{O}$-notations.

$$T(m, d) = \begin{cases} Cm & \text{if } d = 1 \\ Cd & \text{if } d > 1 \text{ and } m = 0 \\ Cd + T(m - 1, d) + \\ \frac{d}{m}(Cdm + T(m - 1, d - 1)) & \text{otw.} \end{cases}$$

Note that $T(m, d)$ denotes the expected running time.

---

# 8 Seidels LP-algorithm

Let $C$ be the largest constant in the $\mathcal{O}$-notations.

We show $T(m, d) \leq C f(d) \max\{1, m\}$.

$d = 1$:

$T(m, 1) \leq Cm \leq C f(1) \max\{1, m\}$ for $f(1) \geq 1$

$d > 1; m = 0$:

$T(0, d) \leq \mathcal{O}(d) \leq Cd \leq C f(d) \max\{1, m\}$ for $f(d) \geq d$

$d > 1; m = 1$:

$T(1, d) = \mathcal{O}(d) + T(0, d) + d\big(\mathcal{O}(d) + T(0, d - 1)\big)$

$\leq Cd + Cd + Cd^2 + dC f(d - 1)$

$\leq C f(d) \max\{1, m\}$ for $f(d) \geq 3d^2 + df(d - 1)$

# 8 Seidels LP-algorithm

$d > 1; m > 1:$

(by induction hypothesis statm. true for $d' < d, m' \geq 0$;
and for $d' = d, m' < m$)

$$T(m, d) = \mathcal{O}(d) + T(m-1, d) + \frac{d}{m}\left(\mathcal{O}(dm) + T(m-1, d-1)\right)$$

$$\leq Cd + Cf(d)(m-1) + Cd^2 + \frac{d}{m}Cf(d-1)(m-1)$$

$$\leq 2Cd^2 + Cf(d)(m-1) + dCf(d-1)$$

$$\leq Cf(d)m$$

if $f(d) \geq df(d-1) + 2d^2$.

# 8 Seidels LP-algorithm

▶ Define $f(1) = 3 \cdot 1^2$ and $f(d) = df(d-1) + 3d^2$ for $d > 1$.

Then

$$f(d) = 3d^2 + df(d-1)$$

$$= 3d^2 + d\left[3(d-1)^2 + (d-1)f(d-2)\right]$$

$$= 3d^2 + d\left[3(d-1)^2 + (d-1)\left[3(d-2)^2 + (d-2)f(d-3)\right]\right]$$

$$= 3d^2 + 3d(d-1)^2 + 3d(d-1)(d-2)^2 + \ldots$$

$$\quad + 3d(d-1)(d-2) \cdot \ldots \cdot 4 \cdot 3 \cdot 2 \cdot 1^2$$

$$= 3d!\left(\frac{d^2}{d!} + \frac{(d-1)^2}{(d-1)!} + \frac{(d-2)^2}{(d-2)!} + \ldots\right)$$

$$= \mathcal{O}(d!)$$

since $\sum_{i \geq 1} \frac{i^2}{i!}$ is a constant.

# Complexity

**LP Feasibility Problem (LP feasibility)**

▶ Given $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$. Does there exist $x \in \mathbb{R}$ with $Ax = b$, $x \geq 0$?

▶ Note that allowing $A, b$ to contain rational numbers does not make a difference, as we can multiply every number by a suitable large constant so that everything becomes integral but the feasible region does not change.

**Is this problem in NP or even in P?**

# The Bit Model

**Input size**

▶ The number of bits to represent a number $a \in \mathbb{Z}$ is

$$\lceil \log_2(|a|) \rceil + 1$$

▶ Let for an $m \times n$ matrix $M$, $L(M)$ denote the number of bits required to encode all the numbers in $M$.

$$L(M) := \sum_{i,j} \lceil \log_2(|m_{ij}|) + 1 \rceil$$

▶ In the following we assume that input matrices are encoded in a standard way, where each number is encoded in binary and then suitable separators are added in order to separate distinct number from each other.

▶ Then the input length is $\Theta(L([A|b]))$.

- In the following we sometimes refer to $L := L([A|b])$ as the input size (even though the real input size is something in $\Theta(L([A|b]))$).
- In order to show that LP-decision is in NP we show that if there is a solution $x$ then there exists a small solution for which feasibility can be verified in polynomial time (polynomial in $L([A|b])$).

Suppose that $Ax = b$; $x \geq 0$ is feasible.

Then there exists a basic feasible solution. This means a set $B$ of basic variables such that

$$x_B = A_B^{-1} b$$

and all other entries in $x$ are $0$.

## Size of a Basic Feasible Solution

### Lemma 47
*Let $M \in \mathbb{Z}^{m \times m}$ be an invertable matrix and let $b \in \mathbb{Z}^m$. Further define $L' = L([M \mid b]) + n \log_2 n$. Then a solution to $Mx = b$ has rational components $x_j$ of the form $\frac{D_j}{D}$, where $|D_j| \leq 2^{L'}$ and $|D| \leq 2^{L'}$.*

**Proof:**
Cramers rules says that we can compute $x_j$ as

$$x_j = \frac{\det(M_j)}{\det(M)}$$

where $M_j$ is the matrix obtained from $M$ by replacing the $j$-th column by the vector $b$.

## Bounding the Determinant

Let $X = A_B$. Then

$$|\det(X)| = \left| \sum_{\pi \in S_n} \mathrm{sgn}(\pi) \prod_{1 \leq i \leq n} X_{i\pi(i)} \right|$$
$$\leq \sum_{\pi \in S_n} \prod_{1 \leq i \leq n} |X_{i\pi(i)}|$$
$$\leq n! \cdot 2^{L([A|b])} \leq n^n 2^L \leq 2^{L'} .$$

Analogously for $\det(M_j)$.

This means if $Ax = b$, $x \geq 0$ is feasible we only need to consider vectors $x$ where an entry $x_j$ can be represented by a rational number with encoding length polynomial in the input length $L$.

Hence, the $x$ that we have to guess is of length polynomial in the input-length $L$.

For a given vector $x$ of polynomial length we can check for feasibility in polynomial time.

Hence, LP feasibility is in NP.

## Reducing LP-solving to LP decision.

Given an LP $\max\{c^T x \mid Ax = b; x \geq 0\}$ do a binary search for the optimum solution

(Add constraint $c^T x - \delta = M$; $\delta \geq 0$ or ($c^T x \geq M$). Then checking for feasibility shows whether optimum solution is larger or smaller than $M$).

If the LP is feasible then the binary search finishes in at most

$$\log_2 \left( \frac{2n2^{2L'}}{1/2^{L'}} \right) = \mathcal{O}(L') \ ,$$

as the range of the search is at most $-n2^{2L'}, \ldots, n2^{2L'}$ and the distance between two adjacent values is at least $\frac{1}{\det(A)} \geq \frac{1}{2^{L'}}$.

Here we use $L' = L([A \mid b \mid c]) + n \log_2 n$ (it also includes the encoding size of $c$).

**How do we detect whether the LP is unbounded?**

Let $M_{\max} = n2^{2L'}$ be an upper bound on the objective value of a basic feasible solution.

We can add a constraint $c^T x \geq M_{\max} + 1$ and check for feasibility.

## Ellipsoid Method

- Let $K$ be a convex set.
- Maintain ellipsoid $E$ that is guaranteed to contain $K$ provided that $K$ is non-empty.
- If center $z \in K$ STOP.
- Otw. find a hyperplane separating $K$ from $z$ (e.g. a violated constraint in the LP).
- Shift hyperplane to contain node $z$. $H$ denotes half-space that contains $K$.
- Compute (smallest) ellipsoid $E'$ that contains $K \cap H$.
- REPEAT
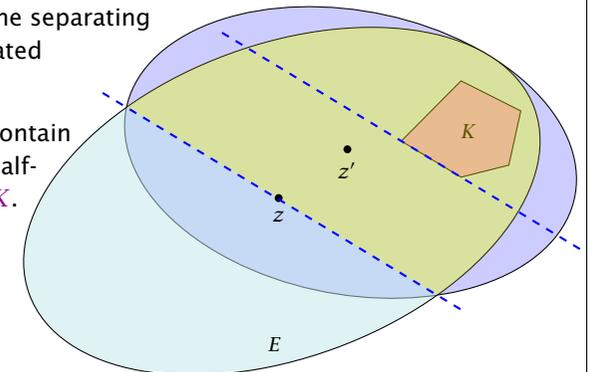
**Issues/Questions:**

- How do you choose the first Ellipsoid? What is its volume?
- What if the polytop $K$ is unbounded?
- How do you measure progress? By how much does the volume decrease in each iteration?
- When can you stop? What is the minimum volume of a non-empty polytop?

**Definition 48**

A mapping $f : \mathbb{R}^n \to \mathbb{R}^n$ with $f(x) = Lx + t$, where $L$ is an invertible matrix is called an affine transformation.

**Definition 49**

A ball in $\mathbb{R}^n$ with center $c$ and radius $r$ is given by

$$B(c, r) = \{x \mid (x - c)^T (x - c) \le r^2\}$$
$$= \{x \mid \sum_i (x - c)_i^2 / r^2 \le 1\}$$

$B(0, 1)$ is called the unit ball.

**Definition 50**

An affine transformation of the unit ball is called an ellipsoid.

From $f(x) = Lx + t$ follows $x = L^{-1}(f(x) - t)$.

$$f(B(0, 1)) = \{f(x) \mid x \in B(0, 1)\}$$
$$= \{y \in \mathbb{R}^n \mid L^{-1}(y - t) \in B(0, 1)\}$$
$$= \{y \in \mathbb{R}^n \mid (y - t)^T L^{-1^T} L^{-1}(y - t) \le 1\}$$
$$= \{y \in \mathbb{R}^n \mid (y - t)^T Q^{-1}(y - t) \le 1\}$$

where $Q = LL^T$ is an invertible matrix.

## How to Compute the New Ellipsoid

- ▶ Use $f^{-1}$ (recall that $f = Lx + t$ is the affine transformation of the unit ball) to rotate/distort the ellipsoid (back) into the unit ball.

- ▶ Use a rotation $R^{-1}$ to rotate the unit ball such that the normal vector of the halfspace is parallel to $e_1$.

- ▶ Compute the new center $\hat{c}'$ and the new matrix $\hat{Q}'$ for this simplified setting.

- ▶ Use the transformations $R$ and $f$ to get the new center $c'$ and the new matrix $Q'$ for the original ellipsoid $E$.

---
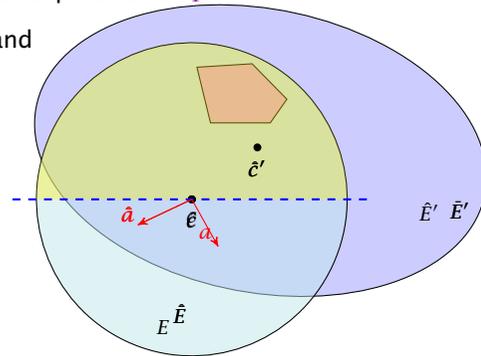
## The Easy Case
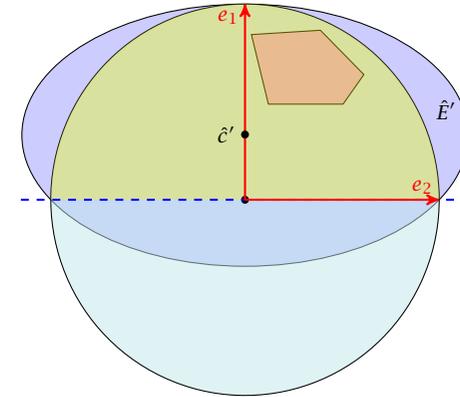
- ▶ The new center lies on axis $x_1$. Hence, $\hat{c}' = te_1$ for $t > 0$.
- ▶ The vectors $e_1, e_2, \ldots$ have to fulfill the ellipsoid constraint with equality. Hence $(e_i - \hat{c}')^T \hat{Q}'^{-1} (e_i - \hat{c}') = 1$.

---

## The Easy Case

- ▶ The obtain the matrix $\hat{Q}'^{-1}$ for our ellipsoid $\hat{E}'$ note that $\hat{E}'$ is axis-parallel.

- ▶ Let $a$ denote the radius along the $x_1$-axis and let $b$ denote the (common) radius for the other axes.

- ▶ The matrix

$$\hat{L}' = \begin{pmatrix} a & 0 & \cdots & 0 \\ 0 & b & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & b \end{pmatrix}$$

  maps the unit ball (via function $\hat{f}'(x) = \hat{L}'x$) to an axis-parallel ellipsoid with radius $a$ in direction $x_1$ and $b$ in all other directions.

---

## The Easy Case

- ▶ As $\hat{Q}' = \hat{L}'\hat{L}'^t$ the matrix $\hat{Q}'^{-1}$ is of the form

$$\hat{Q}'^{-1} = \begin{pmatrix} \frac{1}{a^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{b^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{b^2} \end{pmatrix}$$

## The Easy Case

- $(e_1 - \hat{c}')^T \hat{Q}'^{-1} (e_1 - \hat{c}') = 1$ gives

$$\begin{pmatrix} 1-t \\ 0 \\ \vdots \\ 0 \end{pmatrix}^T \cdot \begin{pmatrix} \frac{1}{a^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{b^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{b^2} \end{pmatrix} \cdot \begin{pmatrix} 1-t \\ 0 \\ \vdots \\ 0 \end{pmatrix} = 1$$

- This gives $(1-t)^2 = a^2$.

---

## The Easy Case

- For $i \neq 1$ the equation $(e_i - \hat{c}')^T \hat{Q}'^{-1} (e_i - \hat{c}') = 1$ gives

$$\begin{pmatrix} -t \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}^T \cdot \begin{pmatrix} \frac{1}{a^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{b^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{b^2} \end{pmatrix} \cdot \begin{pmatrix} -t \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = 1$$

- This gives $\frac{t^2}{a^2} + \frac{1}{b^2} = 1$, and hence

$$\frac{1}{b^2} = 1 - \frac{t^2}{a^2} = 1 - \frac{t^2}{(1-t)^2} = \frac{1-2t}{(1-t)^2}$$

---

## Summary

So far we have

$$a = 1 - t \quad \text{and} \quad b = \frac{1-t}{\sqrt{1-2t}}$$

---

## The Easy Case

We still have many choices for $t$:
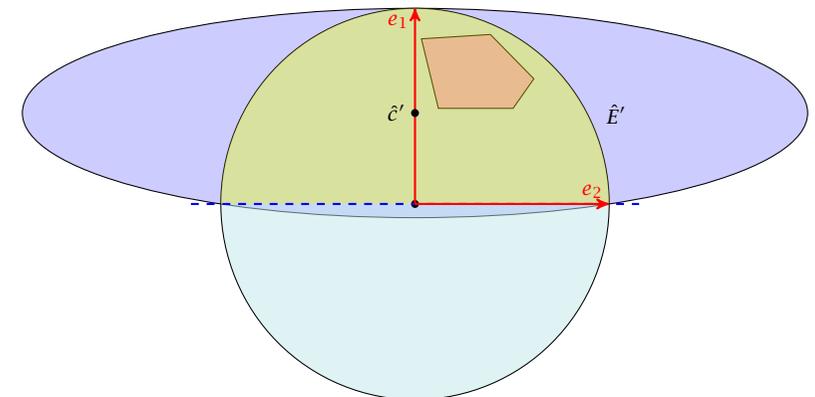


Choose $t$ such that the volume of $\hat{E}'$ is minimal!!!

## The Easy Case

We want to choose $t$ such that the volume of $\hat{E}'$ is minimal.

**Lemma 51**

*Let $L$ be an affine transformation and $K \subseteq \mathbb{R}^n$. Then*

$$\mathrm{vol}(L(K)) = |\det(L)| \cdot \mathrm{vol}(K) \ .$$

## n-dimensional volume
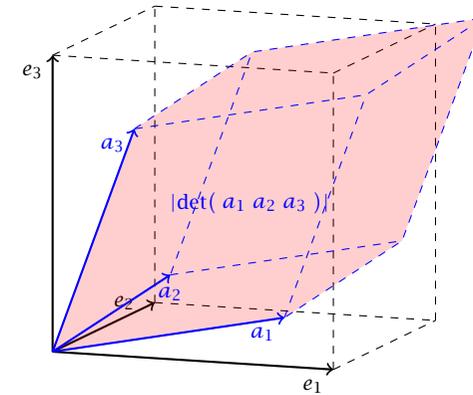
## The Easy Case

▶ We want to choose $t$ such that the volume of $\hat{E}'$ is minimal.

$$\mathrm{vol}(\hat{E}') = \mathrm{vol}(B(0,1)) \cdot |\det(\hat{L}')| \ ,$$

where $\hat{Q}' = \hat{L}'\hat{L}'^{T}$.

▶ We have

$$\hat{L}'^{-1} = \begin{pmatrix} \frac{1}{a} & 0 & \dots & 0 \\ 0 & \frac{1}{b} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{b} \end{pmatrix} \text{ and } \hat{L}' = \begin{pmatrix} a & 0 & \dots & 0 \\ 0 & b & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & b \end{pmatrix}$$

▶ Note that $a$ and $b$ in the above equations depend on $t$, by the previous equations.

## The Easy Case

$$\begin{aligned} \mathrm{vol}(\hat{E}') &= \mathrm{vol}(B(0,1)) \cdot |\det(\hat{L}')| \\ &= \mathrm{vol}(B(0,1)) \cdot ab^{n-1} \\ &= \mathrm{vol}(B(0,1)) \cdot (1-t) \cdot \left(\frac{1-t}{\sqrt{1-2t}}\right)^{n-1} \\ &= \mathrm{vol}(B(0,1)) \cdot \frac{(1-t)^n}{(\sqrt{1-2t})^{n-1}} \end{aligned}$$

## The Easy Case

$$\frac{d\,\mathrm{vol}(\hat{E}')}{dt} = \frac{d}{dt}\left(\frac{(1-t)^n}{(\sqrt{1-2t})^{n-1}}\right)$$

$$= \frac{1}{N^2} \cdot \left( (-1) \cdot n(1-t)^{n-1} \cdot (\sqrt{1-2t})^{n-1} \right.$$

derivative of numerator     denominator

$N$ = denominator

$$\underbrace{-(n-1)(\sqrt{1-2t})^{n-2}}_{\text{outer derivative}} \cdot \underbrace{\frac{1}{2\sqrt{1-2t}} \cdot (-2)}_{\text{inner derivative}} \cdot \underbrace{(1-t)^n}_{\text{numerator}} \Big)$$

$$= \frac{1}{N^2} \cdot (\sqrt{1-2t})^{n-3} \cdot (1-t)^{n-1}$$

$$\cdot \Big( (n-1)(1-t) - n(1-2t) \Big)$$

$$= \frac{1}{N^2} \cdot (\sqrt{1-2t})^{n-3} \cdot (1-t)^{n-1} \cdot \Big( (n+1)t - 1 \Big)$$

---

## The Easy Case

- We obtain the minimum for $t = \frac{1}{n+1}$.
- For this value we obtain

$$a = 1 - t = \frac{n}{n+1} \quad\text{and}\quad b = \frac{1-t}{\sqrt{1-2t}} = \frac{n}{\sqrt{n^2-1}}$$

To see the equation for $b$, observe that

$$b^2 = \frac{(1-t)^2}{1-2t} = \frac{(1 - \frac{1}{n+1})^2}{1 - \frac{2}{n+1}} = \frac{(\frac{n}{n+1})^2}{\frac{n-1}{n+1}} = \frac{n^2}{n^2-1}$$

---

## The Easy Case

Let $\gamma_n = \frac{\mathrm{vol}(\hat{E}')}{\mathrm{vol}(B(0,1))} = ab^{n-1}$ be the ratio by which the volume changes:

$$\gamma_n^2 = \left(\frac{n}{n+1}\right)^2 \left(\frac{n^2}{n^2-1}\right)^{n-1}$$

$$= \left(1 - \frac{1}{n+1}\right)^2 \left(1 + \frac{1}{(n-1)(n+1)}\right)^{n-1}$$

$$\leq e^{-2\frac{1}{n+1}} \cdot e^{\frac{1}{n+1}}$$

$$= e^{-\frac{1}{n+1}}$$

where we used $(1+x)^a \leq e^{ax}$ for $x \in \mathbb{R}$ and $a > 0$.

This gives $\gamma_n \leq e^{-\frac{1}{2(n+1)}}$.

---

## How to Compute the New Ellipsoid

- Use $f^{-1}$ (recall that $f = Lx + t$ is the affine transformation of the unit ball) to rotate/distort the ellipsoid (back) into the unit ball.

- Use a rotation $R^{-1}$ to rotate the unit ball such that the normal vector of the halfspace is parallel to $e_1$.

- Compute the new center $\hat{c}'$ and the new matrix $\hat{Q}'$ for this simplified setting.

- Use the transformations $R$ and $f$ to get the new center $c'$ and the new matrix $Q'$ for the original ellipsoid $E$.
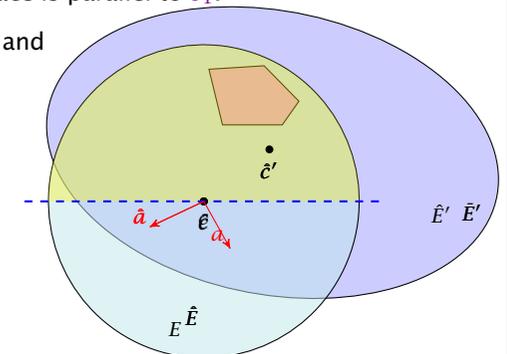
**Our progress is the same:**

$$e^{-\frac{1}{2(n+1)}} \geq \frac{\mathrm{vol}(\hat{E}')}{\mathrm{vol}(B(0,1))} = \frac{\mathrm{vol}(\hat{E}')}{\mathrm{vol}(\hat{E})} = \frac{\mathrm{vol}(R(\hat{E}'))}{\mathrm{vol}(R(\hat{E}))}$$

$$= \frac{\mathrm{vol}(\bar{E}')}{\mathrm{vol}(\bar{E})} = \frac{\mathrm{vol}(f(\bar{E}'))}{\mathrm{vol}(f(\bar{E}))} = \frac{\mathrm{vol}(E')}{\mathrm{vol}(E)}$$

Here it is important that mapping a set with affine function $f(x) = Lx + t$ changes the volume by factor $\det(L)$.

## The Ellipsoid Algorithm

**How to Compute The New Parameters?**

The transformation function of the (old) ellipsoid: $f(x) = Lx + c$;

The halfspace to be intersected: $H = \{x \mid a^T(x - c) \leq 0\}$;

$$\begin{aligned} f^{-1}(H) &= \{f^{-1}(x) \mid a^T(x - c) \leq 0\} \\ &= \{f^{-1}(f(y)) \mid a^T(f(y) - c) \leq 0\} \\ &= \{y \mid a^T(f(y) - c) \leq 0\} \\ &= \{y \mid a^T(Ly + c - c) \leq 0\} \\ &= \{y \mid (a^T L)y \leq 0\} \end{aligned}$$

This means $\bar{a} = L^T a$.

## The Ellipsoid Algorithm

After rotating back (applying $R^{-1}$) the normal vector of the halfspace points in negative $x_1$-direction. Hence,

$$R^{-1}\left(\frac{L^T a}{\|L^T a\|}\right) = -e_1 \quad \Rightarrow \quad -\frac{L^T a}{\|L^T a\|} = R \cdot e_1$$

Hence,

$$\bar{c}' = R \cdot \hat{c}' = R \cdot \frac{1}{n+1} e_1 = -\frac{1}{n+1} \frac{L^T a}{\|L^T a\|}$$

$$\begin{aligned} c' = f(\bar{c}') &= L \cdot \bar{c}' + c \\ &= -\frac{1}{n+1} L \frac{L^T a}{\|L^T a\|} + c \\ &= c - \frac{1}{n+1} \frac{Qa}{\sqrt{a^T Q a}} \end{aligned}$$

For computing the matrix $Q'$ of the new ellipsoid we assume in the following that $\hat{E}', \bar{E}'$ and $E'$ refer to the ellispoids centered in the origin.

Recall that

$$\hat{Q}' = \begin{pmatrix} a^2 & 0 & \dots & 0 \\ 0 & b^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & b^2 \end{pmatrix}$$

This gives

$$\hat{Q}' = \frac{n^2}{n^2 - 1}\Big(I - \frac{2}{n+1}e_1 e_1^T\Big)$$

because for $a = n/n+1$ and $b = n/\sqrt{n^2-1}$

$$b^2 - b^2 \frac{2}{n+1} = \frac{n^2}{n^2 - 1} - \frac{2n^2}{(n-1)(n+1)^2}$$

$$= \frac{n^2(n+1) - 2n^2}{(n-1)(n+1)^2} = \frac{n^2(n-1)}{(n-1)(n+1)^2} = a^2$$

$$\bar{E}' = R(\hat{E}')$$
$$= \{R(x) \mid x^T \hat{Q}'^{-1} x \le 1\}$$
$$= \{y \mid (R^{-1}y)^T \hat{Q}'^{-1} R^{-1} y \le 1\}$$
$$= \{y \mid y^T (R^T)^{-1} \hat{Q}'^{-1} R^{-1} y \le 1\}$$
$$= \{y \mid y^T \underbrace{(R\hat{Q}'R^T)}_{\bar{Q}'}^{-1} y \le 1\}$$

# 9 The Ellipsoid Algorithm

Hence,

$$\bar{Q}' = R\hat{Q}'R^T$$
$$= R \cdot \frac{n^2}{n^2 - 1}\Big(I - \frac{2}{n+1}e_1 e_1^T\Big) \cdot R^T$$
$$= \frac{n^2}{n^2 - 1}\Big(R \cdot R^T - \frac{2}{n+1}(Re_1)(Re_1)^T\Big)$$
$$= \frac{n^2}{n^2 - 1}\Big(I - \frac{2}{n+1}\frac{L^T a a^T L}{\|L^T a\|^2}\Big)$$

# 9 The Ellipsoid Algorithm

$$E' = L(\bar{E}')$$
$$= \{L(x) \mid x^T \bar{Q}'^{-1} x \le 1\}$$
$$= \{y \mid (L^{-1}y)^T \bar{Q}'^{-1} L^{-1} y \le 1\}$$
$$= \{y \mid y^T (L^T)^{-1} \bar{Q}'^{-1} L^{-1} y \le 1\}$$
$$= \{y \mid y^T \underbrace{(L\bar{Q}'L^T)}_{Q'}^{-1} y \le 1\}$$

## 9 The Ellipsoid Algorithm

Hence,

$$Q' = L\bar{Q}'L^T$$

$$= L \cdot \frac{n^2}{n^2 - 1}\Big(I - \frac{2}{n+1}\frac{L^T a a^T L}{a^T Q a}\Big) \cdot L^T$$

$$= \frac{n^2}{n^2 - 1}\Big(Q - \frac{2}{n+1}\frac{Q a a^T Q}{a^T Q a}\Big)$$

## Incomplete Algorithm

**Algorithm 1** ellipsoid-algorithm

1: **input:** point $c \in \mathbb{R}^n$, convex set $K \subseteq \mathbb{R}^n$
2: **output:** point $x \in K$ or "$K$ is empty"
3: $Q \leftarrow$ ???
4: **repeat**
5:     **if** $c \in K$ **then return** $c$
6:     **else**
7:         choose a violated hyperplane $a$
8:         $c \leftarrow c - \dfrac{1}{n+1}\dfrac{Q a}{\sqrt{a^T Q a}}$
9:         $Q \leftarrow \dfrac{n^2}{n^2 - 1}\Big(Q - \dfrac{2}{n+1}\dfrac{Q a a^T Q}{a^T Q a}\Big)$
10:     **endif**
11: **until** ???
12: **return** "$K$ is empty"

## Repeat: Size of basic solutions

**Lemma 52**

*Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ be a bounded polyhedron. Let $\langle a_{\max} \rangle$ be the maximum encoding length of an entry in $A, b$. Then every entry $x_j$ in a basic solution fulfills $|x_j| = \frac{D_j}{D}$ with $D_j, D \leq 2^{2n\langle a_{\max}\rangle + 2n\log_2 n}$.*

In the following we use $\delta := 2^{2n\langle a_{\max}\rangle + 2n\log_2 n}$.

Note that here we have $P = \{x \mid Ax \leq b\}$. The previous lemmas we had about the size of feasible solutions were slightly different as they were for different polytopes.

## Repeat: Size of basic solutions

**Proof:**

Let $\bar{A} = \begin{bmatrix} A & -A & I_m \\ -A & A \end{bmatrix}$, $\bar{b} = \begin{pmatrix} b \\ -b \end{pmatrix}$, be the matrix and right-hand vector after transforming the system to standard form.

The determinant of the matrices $\bar{A}_B$ and $\bar{M}_j$ (matrix obt. when replacing the $j$-th column of $\bar{A}_B$ by $\bar{b}$) can become at most

$$\det(\bar{A}_B), \det(\bar{M}_j) \leq \|\vec{\ell}_{\max}\|^{2n}$$
$$\leq (\sqrt{2n} \cdot 2^{\langle a_{\max}\rangle})^{2n} \leq 2^{2n\langle a_{\max}\rangle + 2n\log_2 n},$$

where $\vec{\ell}_{\max}$ is the longest column-vector that can be obtained after deleting all but $2n$ rows and columns from $\bar{A}$.

This holds because columns from $I_m$ selected when going from $\bar{A}$ to $\bar{A}_B$ do not increase the determinant. Only the at most $2n$ columns from matrices $A$ and $-A$ that $\bar{A}$ consists of contribute.

## How do we find the first ellipsoid?

For feasibility checking we can assume that the polytop $P$ is bounded; it is sufficient to consider basic solutions.

Every entry $x_i$ in a basic solution fulfills $|x_i| \leq \delta$.

Hence, $P$ is contained in the cube $-\delta \leq x_i \leq \delta$.

A vector in this cube has at most distance $R := \sqrt{n}\delta$ from the origin.

Starting with the ball $E_0 := B(0, R)$ ensures that $P$ is completely contained in the initial ellipsoid. This ellipsoid has volume at most $R^n B(0, 1) \leq (n\delta)^n B(0, 1)$.

## When can we terminate?

Let $P := \{x \mid Ax \leq b\}$ with $A \in \mathbb{Z}$ and $b \in \mathbb{Z}$ be a bounded polytop. Let $\langle a_{\max} \rangle$ be the encoding length of the largest entry in $A$ or $b$.

Consider the following polyhedron

$$P_\lambda := \left\{ x \mid Ax \leq b + \frac{1}{\lambda} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \right\},$$

where $\lambda = \delta^2 + 1$.

### Lemma 53
$P_\lambda$ is feasible if and only if $P$ is feasible.

$\Longleftarrow$: obvious!

$\Longrightarrow$:
Consider the polyhedrons

$$\bar{P} = \left\{ x \mid \begin{bmatrix} A & -A & I_m \\ -A & A & \end{bmatrix} x = \begin{pmatrix} b \\ -b \end{pmatrix}; x \geq 0 \right\}$$

and

$$\bar{P}_\lambda = \left\{ x \mid \begin{bmatrix} A & -A & I_m \\ -A & A & \end{bmatrix} x = \begin{pmatrix} b \\ -b \end{pmatrix} + \frac{1}{\lambda} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}; x \geq 0 \right\}.$$

$P$ is feasible if and only if $\bar{P}$ is feasible, and $P_\lambda$ feasible if and only if $\bar{P}_\lambda$ feasible.

$\bar{P}_\lambda$ is bounded since $P_\lambda$ and $P$ are bounded.

Let $\bar{A} = \begin{bmatrix} A & -A & I_m \\ -A & A \end{bmatrix}$, and $\bar{b} = \begin{pmatrix} b \\ -b \end{pmatrix}$.

$\bar{P}_\lambda$ feasible implies that there is a basic feasible solution represented by

$$x_B = \bar{A}_B^{-1} \bar{b} + \frac{1}{\lambda} \bar{A}_B^{-1} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

(The other $x$-values are zero)

The only reason that this basic feasible solution is not feasible for $\bar{P}$ is that one of the basic variables becomes negative.

Hence, there exists $i$ with

$$(\bar{A}_B^{-1} \bar{b})_i < 0 \le (\bar{A}_B^{-1} \bar{b})_i + \frac{1}{\lambda} (\bar{A}_B^{-1} \vec{1})_i$$

---

By Cramers rule we get

$$(\bar{A}_B^{-1} \bar{b})_i < 0 \quad \implies \quad (\bar{A}_B^{-1} \bar{b})_i \le -\frac{1}{\det(\bar{A}_B)}$$

and

$$(\bar{A}_B^{-1} \vec{1})_i \le \det(\bar{M}_j) \ ,$$

where $\bar{M}_j$ is obtained by replacing the $j$-th column of $\bar{A}_B$ by $\vec{1}$.

However, we showed that the determinants of $\bar{A}_B$ and $\bar{M}_j$ can become at most $\delta$.

Since, we chose $\lambda = \delta^2 + 1$ this gives a contradiction.

---

**Lemma 54**

*If $P_\lambda$ is feasible then it contains a ball of radius $r := 1/\delta^3$. This has a volume of at least $r^n \text{vol}(B(0,1)) = \frac{1}{\delta^{3n}} \text{vol}(B(0,1))$.*

**Proof:**

If $P_\lambda$ feasible then also $P$. Let $x$ be feasible for $P$.
This means $Ax \le b$.

Let $\vec{\ell}$ with $\|\vec{\ell}\| \le r$. Then

$$(A(x + \vec{\ell}))_i = (Ax)_i + (A\vec{\ell})_i \le b_i + \vec{a}_i^T \vec{\ell}$$
$$\le b_i + \|\vec{a}_i\| \cdot \|\vec{\ell}\| \le b_i + \sqrt{n} \cdot 2^{\langle a_{\max} \rangle} \cdot r$$
$$\le b_i + \frac{\sqrt{n} \cdot 2^{\langle a_{\max} \rangle}}{\delta^3} \le b_i + \frac{1}{\delta^2 + 1} \le b_i + \frac{1}{\lambda}$$

Hence, $x + \vec{\ell}$ is feasible for $P_\lambda$ which proves the lemma.

---

How many iterations do we need until the volume becomes too small?

$$e^{-\frac{i}{2(n+1)}} \cdot \text{vol}(B(0,R)) < \text{vol}(B(0,r))$$

Hence,

$$i > 2(n+1) \ln \left( \frac{\text{vol}(B(0,R))}{\text{vol}(B(0,r))} \right)$$
$$= 2(n+1) \ln \left( n^n \delta^n \cdot \delta^{3n} \right)$$
$$= 8n(n+1) \ln(\delta) + 2(n+1)n \ln(n)$$
$$= \mathcal{O}(\text{poly}(n, \langle a_{\max} \rangle))$$

**Algorithm 1** ellipsoid-algorithm

1: **input:** point $c \in \mathbb{R}^n$, convex set $K \subseteq \mathbb{R}^n$, radii $R$ and $r$
2:      with $K \subseteq B(c, R)$, and $B(x, r) \subseteq K$ for some $x$
3: **output:** point $x \in K$ or "$K$ is empty"
4: $Q \leftarrow \text{diag}(R^2, \ldots, R^2)$ // i.e., $L = \text{diag}(R, \ldots, R)$
5: **repeat**
6:      **if** $c \in K$ **then return** $c$
7:      **else**
8:          choose a violated hyperplane $a$
9:          $c \leftarrow c - \dfrac{1}{n+1} \dfrac{Qa}{\sqrt{a^T Q a}}$
10:          $Q \leftarrow \dfrac{n^2}{n^2 - 1}\Big(Q - \dfrac{2}{n+1} \dfrac{Q a a^T Q}{a^T Q a}\Big)$
11:      **endif**
12: **until** $\det(Q) \leq r^{2n}$ // i.e., $\det(L) \leq r^n$
13: **return** "$K$ is empty"

**Separation Oracle:**

Let $K \subseteq \mathbb{R}^n$ be a convex set. A separation oracle for $K$ is an algorithm $A$ that gets as input a point $x \in \mathbb{R}^n$ and either

- ▶ certifies that $x \in K$,
- ▶ or finds a hyperplane separating $x$ from $K$.

We will usually assume that $A$ is a polynomial-time algorithm.

In order to find a point in $K$ we need

- ▶ a guarantee that a ball of radius $r$ is contained in $K$,
- ▶ an initial ball $B(c, R)$ with radius $R$ that contains $K$,
- ▶ a separation oracle for $K$.

The Ellipsoid algorithm requires $\mathcal{O}(\text{poly}(n) \cdot \log(R/r))$ iterations. Each iteration is polytime for a polynomial-time Separation oracle.

# 10 Karmarkars Algorithm

- ▶ inequalities $Ax \leq b$; $m \times n$ matrix $A$ with rows $a_i^T$
- ▶ $P = \{x \mid Ax \leq b\}$; $P° := \{x \mid Ax < b\}$
- ▶ interior point algorithm: $x \in P°$ throughout the algorithm
- ▶ for $x \in P°$ define

$$s_i(x) := b_i - a_i^T x$$

  as the slack of the $i$-th constraint

**logarithmic barrier function:**

$$\phi(x) = -\sum_{i=1}^{m} \log(s_i(x))$$

Penalty for point $x$; points close to the boundary have a very large penalty.

picture of barrier function

## Gradient and Hessian

**Taylor approximation:**

$$\phi(x + \epsilon) \approx \phi(x) + \nabla\phi(x)^T \epsilon + \frac{1}{2}\epsilon^T \nabla^2 \phi(x)\epsilon$$

**Gradient:**

$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{s_i(x)} \cdot a_i = A^T d_x$$

where $d_x^T = (1/s_1(x), \ldots, 1/s_m(x))$. ($d_x$ vector of inverse slacks)

**Hessian:**

$$H_x := \nabla^2 \phi(x) = \sum_{i=1}^{m} \frac{1}{s_i(x)^2} a_i a_i^T = A^T D_x^2 A$$

with $D_x = \mathrm{diag}(d_x)$.

---

$$\frac{\partial \phi(x)}{\partial x_i} = \frac{\partial}{\partial x_i}\left( -\sum_r w_r \ln(s_r(x)) \right)$$

$$= -\sum_r w_r \frac{\partial}{\partial x_i}\left( \ln(s_r(x)) \right)$$

$$= -\sum_r w_r \frac{1}{s_r(x)} \frac{\partial}{\partial x_i}\left( s_r(x) \right)$$

$$= -\sum_r w_r \frac{1}{s_r(x)} \frac{\partial}{\partial x_i}\left( b_r - a_r^T x \right)$$

$$= \sum_r w_r \frac{1}{s_r(x)} \frac{\partial}{\partial x_i}\left( a_r^T x \right)$$

$$= \sum_r w_r \frac{1}{s_r(x)} A_{ri}$$

The $i$-th entry of the gradient vector is $\sum_r w_r/s_r(x) \cdot A_{ri}$. This gives that the gradient is

$$\nabla\phi(x) = \sum_r w_r/s_r(x)a_r = A^T D_x W \vec{1}$$

---

$$\frac{\partial}{\partial x_j}\left( \sum_r \frac{w_r}{s_r(x)} A_{ri} \right) = \sum_r w_r A_{ri} \left( -\frac{1}{s_r(x)^2} \right) \cdot \frac{\partial}{\partial x_j}\left( s_r(x) \right)$$

$$= \sum_r w_r A_{ri} \frac{1}{s_r(x)^2} A_{rj}$$

Note that $\sum_r A_{ri} A_{rj} = (A^T A)_{ij}$. Adding the additional factors $w_r/s_r(x)^2$ can be done with a diagonal matrix.

Hence the Hessian is

$$H_x = A^T D W D A$$

---

$H_x$ is positive semi-definite for $x \in P^\circ$

$$u^T H_x u = u^T A^T D_x^2 A u = \|D_x A u\|_2^2 \geq 0$$

This gives that $\phi(x)$ is convex.

If $\mathrm{rank}(A) = n$, $H_x$ is positive definite for $x \in P^\circ$

$$u^T H_x u = \|D_x A u\|_2^2 > 0 \text{ for } u \neq 0$$

This gives that $\phi(x)$ is **strictly** convex.

$\|u\|_{H_x} := \sqrt{u^T H_x u}$ is a (semi-)norm; the unit ball w.r.t. this norm is an ellipsoid.

## Dilkin Ellipsoid

$$E_x = \{y \mid (y-x)^T H_x (y-x) \leq 1\} = \{y \mid \|y-x\|_{H_x} \leq 1\}$$

**Points in $E_x$ are feasible!!!**

$$
\begin{aligned}
(y-x)^T H_x (y-x) &= (y-x)^T A^T D_x^2 A (y-x) \\
&= \sum_{i=1}^{m} \frac{(a_i^T (y-x))^2}{s_i(x)^2} \\
&= \sum_{i=1}^{m} \frac{(\text{change of distance to } i\text{-th constraint going from } x \text{ to } y)^2}{(\text{distance of } x \text{ to } i\text{-th constraint})^2} \\
&\leq 1
\end{aligned}
$$

In order to become infeasible when going from $x$ to $y$ one of the terms in the sum would need to be larger than 1.

---

## Dilkin Ellipsoids

---
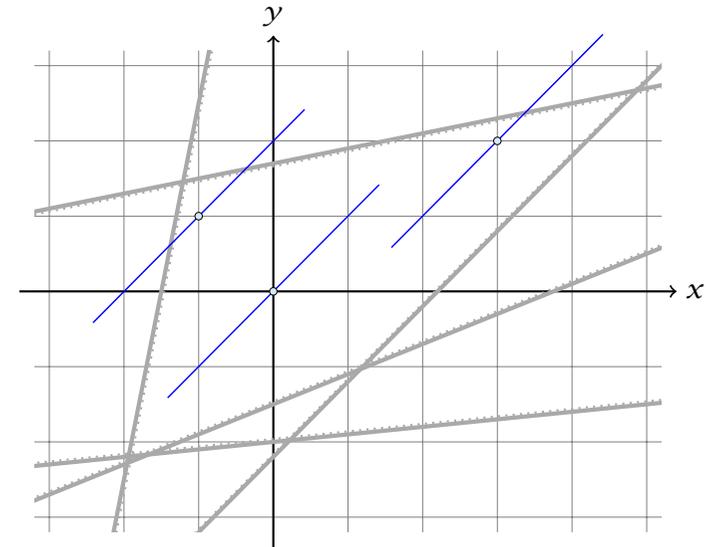
## Analytic Center

$$x_{\text{ac}} := \arg\min_{x \in P^\circ} \phi(x)$$

▸ $x_{\text{ac}}$ is solution to

$$\nabla \phi(x) = \sum_{i=1}^{m} \frac{1}{s_i(x)} a_i = 0$$

▸ depends on the description of the polytope
▸ $x_{\text{ac}}$ exists and is unique iff $P^\circ$ is nonempty and bounded

---

## Central Path

In the following we assume that the LP and its dual are strictly feasible and that $\text{rank}(A) = n$.

**Central Path:**
Set of points $\{x^*(t) \mid t > 0\}$ with

$$x^*(t) = \text{argmin}_x \{t c^T x + \phi(x)\}$$

▸ $t = 0$: analytic center
▸ $t = \infty$: optimum solution

$x^*(t)$ exists and is unique for all $t \geq 0$.

## Slide 1

**primal-dual pair:**

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \le b \end{array}$$

$$\begin{array}{ll} \max & -b^T z \\ \text{s.t.} & A^T z + c = 0 \\ & z \ge 0 \end{array}$$

we assume primal and dual problems are strictly feasible;
$\operatorname{rank}(A) = n$.

## Slide 2

Point $x^*(t)$ on central path is solution to $tc + \nabla\phi(x) = 0$ (force field interpretation).

This means

$$tc + \sum_{i=1}^{m} \frac{1}{b_i - a_i^T x^*(t)} = 0$$

or

$$c + \sum_{i=1}^{m} z_i^*(t) a_i = 0 \ \text{ with } \ z_i^*(t) = \frac{1}{t(b_i - a_i^T x^*(t))}$$

- $z_i^*(t)$ is strictly dual feasible
- duality gap between $x := x^*(t)$ and $z := z^*(t)$ is

$$c^T x + b^T z = (b - Ax)^T z = \frac{m}{t}$$

- if this gap is less than $1/\Omega(2^L)$ we can snap to an optimum point

## Slide 3

## Slide 4
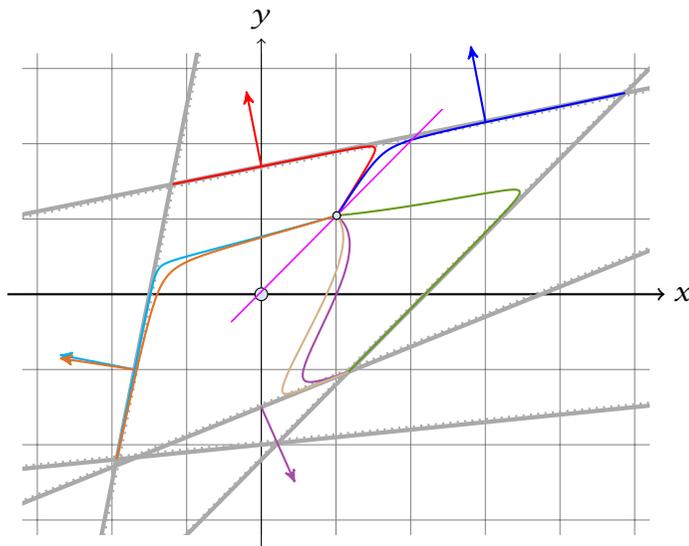
# Path-following Methods

**Algorithm 1** PathFollowing
1: start at analytic center
2: **while** solution not good enough **do**
3:     make step to improve objective function
4:     recenter to return to central path

**Questions/Remarks**

- how do we get to analytic center?
- when is solution "good enough"?
- (usually) improvement step tries to stay feasible, how?
- recentering step should
  - be fast
  - not undo (too much of) improvement

## Centering Problem

$$\text{minimize } f_t(x) = tc^T x + \phi(x)$$

minimizing this gives point $x^*(t)$ on central path

---

**Newton Step** at $x \in P^\circ$

$$
\begin{aligned}
\Delta x_{\mathrm{nt}} &= -H^{-1} \nabla f_t(x) \\
&= -H^{-1}(tc + \nabla \phi(x)) \\
&= -(A^T D_x^2 A)^{-1}(tc + A^T d_x)
\end{aligned}
$$

**Newton Iteration:**

$$x := x + \Delta x_{\mathrm{nt}}$$

---

## Measuring Progress of Newton Step

**Newton decrement:**

$$
\begin{aligned}
\lambda_t(x) &= \| D_x A \Delta x_{\mathrm{nt}} \| \\
&= \| \Delta x_{\mathrm{nt}} \|_{H_x}
\end{aligned}
$$

Square of Newton decrement is linear estimate of reduction if we do a Newton step:

$$-\lambda_t(x)^2 = \nabla f_t(x)^T \Delta x_{\mathrm{nt}}$$

- $\lambda_t(x) = 0$ iff $x = x^*(t)$
- $\lambda_t(x)$ is measure of proximity of $x$ to $x^*(t)$

---

## Convergence of Newtons Method

**Theorem 55**
*If $\lambda_t(x) < 1$ then*

- $x_+ := x + \Delta x_{nt} \in P^\circ$ *(new point feasible)*
- $\lambda_t(x_+) \le \lambda_t(x)^2$

This means we have quadratic convergence. Very fast.

**feasibility:**

- $\lambda_t(x) = \|\Delta x_{nt}\| < 1$; hence $x_+$ lies in the Dilkin ellipsoid around $x$.

---

**bound on $\lambda_t(x^+)$:**

we use $D := D_x = \mathrm{diag}(d_x)$ and $D_+ := D_{x^+} = \mathrm{diag}(d_{x^+})$

$$
\begin{aligned}
\lambda_t(x^+)^2 &= \|D_+ A\Delta x_{nt}^+\|^2 \\
&\leq \|D_+ A\Delta x_{nt}^+\|^2 + \|D_+ A\Delta x_{nt}^+ + (I - D_+^{-1}D)DA\Delta x_{nt}\|^2 \\
&= \|(I - D_+^{-1}D)DA\Delta x_{nt}\|^2
\end{aligned}
$$

To see the last equality

$$
\begin{aligned}
|a^2\| + \|a + b\|^2 &= a^T a + (a^T + b^T)(a + b) \\
&= (a^T + b^T)a + a^T(a + b) + b^T b = \|b\|^2
\end{aligned}
$$

if $a^T(a + b) = 0$.

---

$$
\begin{aligned}
DA\Delta x_{nt} &= DA(x^+ - x) \\
&= D(b - Ax - (b - Ax^+)) \\
&= D(D^{-1}\vec{1} + D^{-1}\vec{1}) \\
&= (I - D_+^{-1}D)\vec{1}
\end{aligned}
$$

$$
\begin{aligned}
a^T(a + b) &\\
&= \Delta x_{nt}^{+T} A^T D_+\left(D_+ A\Delta x_{nt}^+ + (I - D_+^{-1}D)DA\Delta x_{nt}\right) \\
&= \Delta x_{nt}^{+T}\left(A^T D_+^2 A\Delta x_{nt}^+ - A^T D^2 A\Delta x_{nt} + A^T D_+ DA\Delta x_{nt}\right) \\
&= \Delta x_{nt}^{+T}\left(H_+ \Delta x_{nt}^+ - H\Delta x_{nt} + A^T D_+ \vec{1} - A^T D\vec{1}\right) \\
&= \Delta x_{nt}^{+T}\left(-\nabla f_t(x^+) + \nabla f_t(x) + A^T D_+ \vec{1} - A^T D\vec{1}\right) \\
&= 0
\end{aligned}
$$

---

$$
\begin{aligned}
DA\Delta x_{nt} &= DA(x^+ - x) \\
&= D(b - Ax - (b - Ax^+)) \\
&= D(D^{-1}\vec{1} + D^{-1}\vec{1}) \\
&= (I - D_+^{-1}D)\vec{1}
\end{aligned}
$$

$$
\begin{aligned}
a^T(a + b) &\\
&= \Delta x_{nt}^{+T} A^T D_+ \sqrt{W}\left(\sqrt{W}D_+ A\Delta x_{nt}^+ + (I - D_+^{-1}D)\sqrt{W}DA\Delta x_{nt}\right) \\
&= \Delta x_{nt}^{+T}\left(A^T D_+ W D_+ A\Delta x_{nt}^+ - A^T DWDA\Delta x_{nt} + A^T D_+ WDA\Delta x_{nt}\right) \\
&= \Delta x_{nt}^{+T}\left(H_+ \Delta x_{nt}^+ - H\Delta x_{nt} + A^T D_+ W\vec{1} - A^T DW\vec{1}\right) \\
&= \Delta x_{nt}^{+T}\left(-\nabla f_t(x^+) + \nabla f_t(x) + A^T D_+ W\vec{1} - A^T DW\vec{1}\right) \\
&= 0
\end{aligned}
$$

**bound on $\lambda_t(x^+)$:**

we use $D := D_x = \mathrm{diag}(d_x)$ and $D_+ := D_{x^+} = \mathrm{diag}(d_{x^+})$

$$
\begin{aligned}
\lambda_t(x^+)^2 &= \|D_+ A \Delta x_{\mathsf{nt}}^+\|^2 \\
&\le \|D_+ A \Delta x_{\mathsf{nt}}^+\|^2 + \|D_+ A \Delta x_{\mathsf{nt}}^+ + (I - D_+^{-1}D)DA\Delta x_{\mathsf{nt}}\|^2 \\
&= \|(I - D_+^{-1}D)DA\Delta x_{\mathsf{nt}}\|^2 \\
&= \|(I - D_+^{-1}D)^2 \vec{1}\|^2 \\
&\le \|(I - D_+^{-1}D)\vec{1}\|^4 \\
&= \|DA\Delta x_{\mathsf{nt}}\|^4 \\
&= \lambda_t(x)^4
\end{aligned}
$$

The second inequality follows from $\sum_i y_i^4 \le \left(\sum_i y_i^2\right)^2$

---

# Short step barrier method

**simplifying assumptions:**

- a first central point $x^*(t_0)$ is given
- $x^*(t)$ is computed exactly in each iteration

$\epsilon$ is approximation we are aiming for

start at $t = t_0$, repeat until $m/t \le \epsilon$

- compute $x^*(\mu t)$ using Newton starting from $x^*(t)$
- $t := \mu t$

where $\mu = 1 + 1/(2\sqrt{m})$

---

gradient of $f_{t^+}$ at $(x = x^*(t))$

$$
\begin{aligned}
\nabla f_{t^+}(x) &= \nabla f_t(x) + (\mu - 1)tc \\
&= -(\mu - 1)A^T D_x \vec{1}
\end{aligned}
$$

This holds because $0 = \nabla f_t(x) = tc + A^T D_x \vec{1}$.

The Newton decrement is

$$
\begin{aligned}
\lambda_{t^+}(x)^2 &= (\nabla f_{t^+}(x))^T H^{-1} \nabla f_{t^+}(x) \\
&= (\mu - 1)^2 \vec{1}^T B(B^T B)^{-1} B^T \vec{1} \qquad B = D_x A \\
&\le (\mu - 1)^2 m \\
&= 1/4
\end{aligned}
$$

This means we are in the range of quadratic convergence!!!

---

# Number of Iterations

the number of Newton iterations per outer iteration is very small; in practise only 1 or 2

**Number of outer iterations:**

We need $t_k = \mu^k t_0 \ge m/\epsilon$. This holds when

$$
k \ge \frac{\log(m/(\epsilon t_0))}{\log(\mu)}
$$

We get a bound of

$$
\mathcal{O}\!\left(\sqrt{m} \log \frac{m}{\epsilon t_0}\right)
$$

We show how to get a starting point with $t_0 = 1/2^L$. Together with $\epsilon \approx 2^L$ we get $\mathcal{O}(L\sqrt{m})$ iterations.

## How to start...

a damped Newton iteration goes in the direction of $\Delta x_{\text{nt}}$ but only so far as to not leave the polytope;

### Lemma 56 (without proof)

*A damped Newton iteration starting at $x_0$ reaches a point with $\lambda_t(x) \leq \delta$ after*

$$\frac{f_t(x_0) - \min_y f_t(y)}{0.09} + \mathcal{O}(\log\log(1/\delta))$$

*iterations.*

This will allow us to quickly reach a point on the central path ($t \approx 2^L$) when starting very close to it (e.g. at the analytic center).

## How to get close to analytic center?

Let $P = \{Ax \leq b\}$ be our (feasible) polyhedron, and $x_0$ a feasible point.

We change $b \rightarrow b + \frac{1}{\lambda} \cdot \vec{1}$, where $L = L(A, b)$ (encoding length) and $\lambda = 2^{2L}$. Recall that a basis is feasible in the old LP iff it is feasible in the new LP.

After, re-normalizing $A$ and $b$ (for integrality) we have that the point $x_0$ is at distance at least $1$ from every constraint.

The determinant of the matrix $A_B$ for a basis $B$ went up by a factor of $2^{2nL}$.

## How to reach the analytic center?

Start at $x_0$.

Choose $c' := -\nabla\phi(x)$.

$x_0 = x^*(1)$ is point on central path for $c'$ and $t = 1$.

You can travel the central path in both directions. Go towards $0$ until $t \approx 1/2^{nL}$. This requires $\sqrt{m}nL$ outer iterations.

Let $x_{c'}$ denote this point.

Let $x_c$ denote the point that minimizes

$$t \cdot c^T x + \phi(x)$$

(i.e., same value for $t$ but different $c$, hence, different central path).

$$
\begin{aligned}
t \cdot c^T x_{\hat{c}} + \phi(x_{\hat{c}}) &\leq t \cdot c^T x_{\hat{c}} + \phi(x_{\hat{c}}) + t \cdot \hat{c}^T x_{\hat{c}} \\
&\leq t \cdot c^T x_{\hat{c}} + \phi(x_c) + t \cdot \hat{c}^T x_c \\
&\leq t \cdot c^T x_c + \phi(x_c) + t \cdot \left(c^T x_{\hat{c}} + \hat{c}^T x_c\right) \\
&\leq t \cdot c^T x_c + \phi(x_c) + 2t 2^{\langle c_{\max}\rangle} 2^{nL}
\end{aligned}
$$

Choosing $t = 1/2^{\Omega(nL)}$ gives that the last term becomes very small. Hence, using damped Newton we can move to a point on the new central path (for $c$) quickly.

In total for this analysis we require $\mathcal{O}(\sqrt{m}nL)$ outer iterations for the whole algorithm.

One interation can be implemented in $\tilde{\mathcal{O}}(m^3)$ time.

# Part III

# Approximation Algorithms

---

There are many practically important optimization problems that are NP-hard.

**What can we do?**

- Heuristics.
- Exploit special structure of instances occurring in practise.
- Consider algorithms that do not compute the optimal solution but provide solutions that are close to optimum.

---

**Definition 57**

An $\alpha$-approximation for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of $\alpha$ of the value of an optimal solution.

---

**Why approximation algorithms?**

- We need algorithms for hard problems.
- It gives a rigorous mathematical base for studying heuristics.
- It provides a metric to compare the difficulty of various optimization problems.
- Proving theorems may give a deeper theoretical understanding which in turn leads to new algorithmic approaches.

**Why not?**

- Sometimes the results are very pessimistic due to the fact that an algorithm has to provide a close-to-optimum solution on every instance.

**Definition 58**

An optimization problem $P = (\mathcal{I}, \text{sol}, m, \text{goal})$ is in **NPO** if

- $x \in \mathcal{I}$ can be decided in polynomial time
- $y \in \text{sol}(\mathcal{I})$ can be verified in polynomial time
- $m$ can be computed in polynomial time
- $\text{goal} \in \{\min, \max\}$

In other words: the decision problem is there a solution $y$ with $m(x, y)$ at most/at least $z$ is in NP.

- $x$ is problem instance
- $y$ is candidate solution
- $m^*(x)$ cost/profit of an optimal solution

**Definition 59 (Performance Ratio)**

$$R(x, y) := \max\left\{\frac{m(x, y)}{m^*(x)}, \frac{m^*(x)}{m(x, y)}\right\}$$

**Definition 60 ($r$-approximation)**

An algorithm $A$ is an $r$-approximation algorithm iff

$$\forall x \in \mathcal{I} : R(x, A(x)) \le r \ ,$$

and $A$ runs in polynomial time.

**Definition 61 (PTAS)**

A PTAS for a problem $P$ from NPO is an algorithm that takes as input $x \in \mathcal{I}$ and $\epsilon > 0$ and produces a solution $y$ for $x$ with

$$R(x, y) \le 1 + \epsilon \ .$$

The running time is polynomial in $|x|$.

approximation with arbitrary good factor... fast?

**Problems that have a PTAS**

**Scheduling.** Given $m$ jobs with known processing times; schedule the jobs on $n$ machines such that the MAKESPAN is minimized.

**Definition 62 (FPTAS)**
An FPTAS for a problem $P$ from NPO is an algorithm that takes as input $x \in \mathcal{I}$ and $\epsilon > 0$ and produces a solution $y$ for $x$ with

$$R(x, y) \leq 1 + \epsilon \ .$$

The running time is polynomial in $|x|$ and $1/\epsilon$.

approximation with arbitrary good factor... fast!

**Problems that have an FPTAS**

**KNAPSACK.** Given a set of items with profits and weights choose a subset of total weight at most $W$ s.t. the profit is maximized.

**Definition 63 (APX – approximable)**
A problem $P$ from NPO is in APX if there exist a constant $r \geq 1$ and an $r$-approximation algorithm for $P$.

constant factor approximation...

## Problems that are in APX

**MAXCUT.** Given a graph $G = (V, E)$; partition $V$ into two disjoint pieces $A$ and $B$ s.t. the number of edges between both pieces is maximized.

**MAX-3SAT.** Given a 3CNF-formula. Find an assignment to the variables that satisfies the maximum number of clauses.

## Problems with polylogarithmic approximation guarantees

- ▶ Set Cover
- ▶ Minimum Multicut
- ▶ Sparsest Cut
- ▶ Minimum Bisection

There is an $r$-approximation with $r \leq \mathcal{O}(\log^c(|x|))$ for some constant $c$.

Note that only for some of the above problem a matching lower bound is known.

## There are really difficult problems!

### Theorem 64
*For any constant $\epsilon > 0$ there does not exist an $\Omega(n^{1-\epsilon})$-approximation algorithm for the maximum clique problem on a given graph $G$ with $n$ nodes unless $\mathrm{P} = \mathrm{NP}$.*

Note that an $n$-approximation is trivial.

## There are weird problems!
Asymmetric $k$-Center admits an $\mathcal{O}(\log^* n)$-approximation.

There is no $o(\log^* n)$-approximation to Asymmetric $k$-Center unless $NP \subseteq DTIME(n^{\log\log\log n})$.

Class APX not important in practise.

Instead of saying problem $P$ is in APX one says problem $P$ admits a 4-approximation.

One only says that a problem is APX-hard.

A crucial ingredient for the design and analysis of approximation algorithms is a technique to obtain an upper bound (for maximization problems) or a lower bound (for minimization problems).

Therefore Linear Programs or Integer Linear Programs play a vital role in the design of many approximation algorithms.

### Definition 65
An Integer Linear Program or Integer Program is a Linear Program in which all variables are required to be integral.

### Definition 66
A Mixed Integer Program is a Linear Program in which a subset of the variables are required to be integral.

Many important combinatorial optimization problems can be formulated in the form of an Integer Program.

**Note that solving Integer Programs in general is NP-complete!**

# Set Cover

Given a ground set $U$, a collection of subsets $S_1, \ldots, S_k \subseteq U$, where the $i$-th subset $S_i$ has weight/cost $w_i$. Find a collection $I \subseteq \{1, \ldots, k\}$ such that

$$\forall u \in U \, \exists i \in I : \; u \in S_i \quad \text{(every element is covered)}$$

and

$$\sum_{i \in I} w_i \quad \text{is minimized.}$$

---

# Set Cover

---
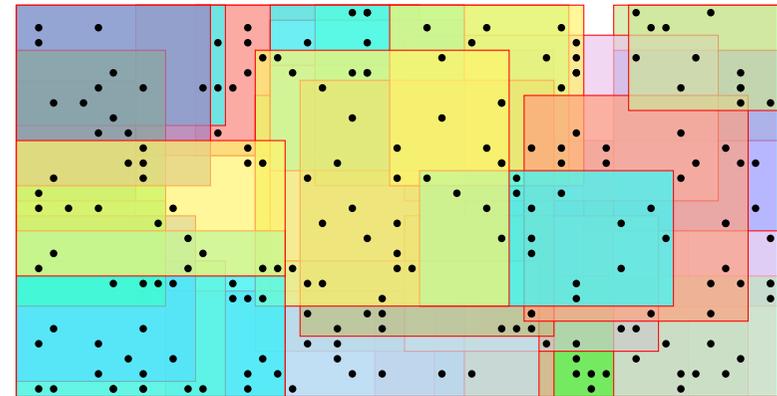
# IP-Formulation of Set Cover

$$
\begin{array}{lrcl}
\min & \sum_i w_i x_i & & \\
\text{s.t.} & \forall u \in U \quad \sum_{i : u \in S_i} x_i & \geq & 1 \\
& \forall i \in \{1, \ldots, k\} \quad x_i & \geq & 0 \\
& \forall i \in \{1, \ldots, k\} \quad x_i & \text{integral} &
\end{array}
$$

---

# Vertex Cover

Given a graph $G = (V, E)$ and a weight $w_v$ for every node. Find a vertex subset $S \subseteq V$ of minimum weight such that every edge is incident to at least one vertex in $S$.

## IP-Formulation of Vertex Cover

$$\begin{array}{lrcl} \min & \sum_{v \in V} w_v x_v & & \\ \text{s.t.} \quad \forall e = (i,j) \in E & x_i + x_j & \geq & 1 \\ \forall v \in V & x_v & \in & \{0,1\} \end{array}$$

## Maximum Weighted Matching

Given a graph $G = (V,E)$, and a weight $w_e$ for every edge $e \in E$. Find a subset of edges of maximum weight such that no vertex is incident to more than one edge.

$$\begin{array}{lrcl} \max & \sum_{e \in E} w_e x_e & & \\ \text{s.t.} \quad \forall v \in V & \sum_{e:v \in e} x_e & \leq & 1 \\ \forall e \in E & x_e & \in & \{0,1\} \end{array}$$

## Maximum Independent Set

Given a graph $G = (V,E)$, and a weight $w_v$ for every node $v \in V$. Find a subset $S \subseteq V$ of nodes of maximum weight such that no two vertices in $S$ are adjacent.

$$\begin{array}{lrcl} \max & \sum_{v \in V} w_v x_v & & \\ \text{s.t.} \quad \forall e = (i,j) \in E & x_i + x_j & \leq & 1 \\ \forall v \in V & x_v & \in & \{0,1\} \end{array}$$

## Knapsack

Given a set of items $\{1,\ldots,n\}$, where the $i$-th item has weight $w_i$ and profit $p_i$, and given a threshold $K$. Find a subset $I \subseteq \{1,\ldots,n\}$ of items of total weight at most $K$ such that the profit is maximized.

$$\begin{array}{lrcl} \max & \sum_{i=1}^{n} p_i x_i & & \\ \text{s.t.} & \sum_{i=1}^{n} w_i x_i & \leq & K \\ \forall i \in \{1,\ldots,n\} & x_i & \in & \{0,1\} \end{array}$$

## Relaxations

### Definition 67
A linear program LP is a relaxation of an integer program IP if any feasible solution for IP is also feasible for LP and if the objective values of these solutions are identical in both programs.

We obtain a relaxation for all examples by writing $x_i \in [0,1]$ instead of $x_i \in \{0,1\}$.

By solving a relaxation we obtain an upper bound for a maximization problem and a lower bound for a minimization problem.

## Relations

**Maximization Problems:**
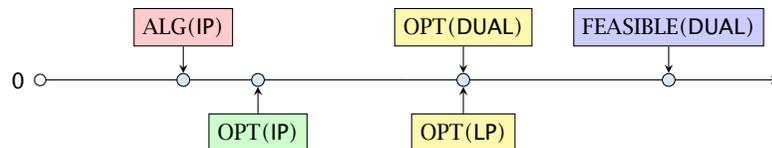


**Minimization Problems:**
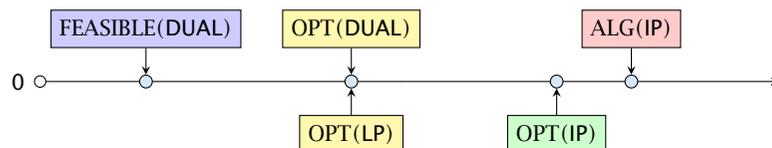
## Technique 1: Round the LP solution.

We first solve the LP-relaxation and then we round the fractional values so that we obtain an integral solution.

**Set Cover relaxation:**

$$\begin{array}{lll}
\min & & \sum_{i=1}^{k} w_i x_i \\
\text{s.t.} & \forall u \in U & \sum_{i:u \in S_i} x_i \geq 1 \\
& \forall i \in \{1,\ldots,k\} & x_i \in [0,1]
\end{array}$$

Let $f_u$ be the number of sets that the element $u$ is contained in (the frequency of $u$). Let $f = \max_u\{f_u\}$ be the maximum frequency.

# Technique 1: Round the LP solution.

**Rounding Algorithm:**
Set all $x_i$-values with $x_i \geq \frac{1}{f}$ to $1$. Set all other $x_i$-values to $0$.

$x_i \geq {}^1/f$.

# Technique 1: Round the LP solution.

**Lemma 68**
*The rounding algorithm gives an $f$-approximation.*

**Proof:** Every $u \in U$ is covered.
▸ We know that $\sum_{i:u \in S_i} x_i \geq 1$.
▸ The sum contains at most $f_u \leq f$ elements.
▸ Therefore one of the sets that contain $u$ must have $x_i \geq {}^1/f$.
▸ This set will be selected. Hence, $u$ is covered.

# Technique 1: Round the LP solution.

The cost of the rounded solution is at most $f \cdot \text{OPT}$.

$$\sum_{i \in I} w_i \leq \sum_{i=1}^{k} w_i (f \cdot x_i)$$
$$= f \cdot \text{cost}(x)$$
$$\leq f \cdot \text{OPT} .$$

# Technique 2: Rounding the Dual Solution.

**Relaxation for Set Cover**

**Primal:**

| min | $\sum_{i \in I} w_i x_i$ |
|---|---|
| s.t. $\forall u$ | $\sum_{i:u \in S_i} x_i \geq 1$ |
| | $x_i \geq 0$ |

**Dual:**

| max | $\sum_{u \in U} y_u$ |
|---|---|
| s.t. $\forall i$ | $\sum_{u:u \in S_i} y_u \leq w_i$ |
| | $y_u \geq 0$ |

## Technique 2: Rounding the Dual Solution.

**Rounding Algorithm:**
Let $I$ denote the index set of sets for which the dual constraint is tight. This means for all $i \in I$

$$\sum_{u:u \in S_i} y_u = w_i$$

## Technique 2: Rounding the Dual Solution.

**Lemma 69**
*The resulting index set is an $f$-approximation.*

**Proof:**
Every $u \in U$ is covered.

▶ Suppose there is a $u$ that is not covered.
▶ This means $\sum_{u:u \in S_i} y_u < w_i$ for all sets $S_i$ that contain $u$.
▶ But then $y_u$ could be increased in the dual solution without violating any constraint. This is a contradiction to the fact that the dual solution is optimal.

## Technique 2: Rounding the Dual Solution.

**Proof:**

$$
\begin{aligned}
\sum_{i \in I} w_i &= \sum_{i \in I} \sum_{u:u \in S_i} y_u \\
&= \sum_u |\{i \in I : u \in S_i\}| \cdot y_u \\
&\leq \sum_u f_u y_u \\
&\leq f \sum_u y_u \\
&\leq f \operatorname{cost}(x^*) \\
&\leq f \cdot \mathrm{OPT}
\end{aligned}
$$

Let $I$ denote the solution obtained by the first rounding algorithm and $I'$ be the solution returned by the second algorithm. Then

$$I \subseteq I' \ .$$

This means $I'$ is never better than $I$.

▶ Suppose that we take $S_i$ in the first algorithm. I.e., $i \in I$.
▶ This means $x_i \geq \frac{1}{f}$.
▶ Because of Complementary Slackness Conditions the corresponding constraint in the dual must be tight.
▶ Hence, the second algorithm will also choose $S_i$.

# Technique 3: The Primal Dual Method

The previous two rounding algorithms have the disadvantage that it is necessary to solve the LP. The following method also gives an $f$-approximation without solving the LP.

For estimating the cost of the solution we only required two properties.

1. The solution is dual feasible and, hence,

$$\sum_u y_u \leq \text{cost}(x^*) \leq \text{OPT}$$

   where $x^*$ is an optimum solution to the primal LP.

2. The set $I$ contains only sets for which the dual inequality is tight.

Of course, we also need that $I$ is a cover.

---

# Technique 3: The Primal Dual Method

**Algorithm 1** PrimalDual
1: $\mathcal{y} \leftarrow 0$
2: $I \leftarrow \emptyset$
3: **while** exists $u \notin \bigcup_{i \in I} S_i$ **do**
4:      increase dual variable $y_u$ until constraint for some new set $S_\ell$ becomes tight
5:      $I \leftarrow I \cup \{\ell\}$

---

# Technique 4: The Greedy Algorithm

**Algorithm 1** Greedy
1: $I \leftarrow \emptyset$
2: $\hat{S}_j \leftarrow S_j$    for all $j$
3: **while** $I$ not a set cover **do**
4:      $\ell \leftarrow \arg\min_{j:\hat{S}_j \neq 0} \frac{w_j}{|\hat{S}_j|}$
5:      $I \leftarrow I \cup \{\ell\}$
6:      $\hat{S}_j \leftarrow \hat{S}_j - S_\ell$    for all $j$

In every round the Greedy algorithm takes the set that covers remaining elements in the most <span style="color:red">cost-effective</span> way.

We choose a set such that the ratio between cost and still uncovered elements in the set is minimized.

---

# Technique 4: The Greedy Algorithm

**Lemma 70**
*Given positive numbers $a_1, \ldots, a_k$ and $b_1, \ldots, b_k$, and $S \subseteq \{1, \ldots, k\}$ then*

$$\min_i \frac{a_i}{b_i} \leq \frac{\sum_{i \in S} a_i}{\sum_{i \in S} b_i} \leq \max_i \frac{a_i}{b_i}$$

## Technique 4: The Greedy Algorithm

Let $n_\ell$ denote the number of elements that remain at the beginning of iteration $\ell$. $n_1 = n = |U|$ and $n_{s+1} = 0$ if we need $s$ iterations.

In the $\ell$-th iteration

$$\min_j \frac{w_j}{|\hat{S}_j|} \le \frac{\sum_{j \in \text{OPT}} w_j}{\sum_{j \in \text{OPT}} |\hat{S}_j|} = \frac{\text{OPT}}{\sum_{j \in \text{OPT}} |\hat{S}_j|} \le \frac{\text{OPT}}{n_\ell}$$

since an optimal algorithm can cover the remaining $n_\ell$ elements with cost OPT.

Let $\hat{S}_j$ be a subset that minimizes this ratio. Hence, $w_j / |\hat{S}_j| \le \frac{\text{OPT}}{n_\ell}$.

## Technique 4: The Greedy Algorithm

Adding this set to our solution means $n_{\ell+1} = n_\ell - |\hat{S}_j|$.

$$w_j \le \frac{|\hat{S}_j| \text{OPT}}{n_\ell} = \frac{n_\ell - n_{\ell+1}}{n_\ell} \cdot \text{OPT}$$

## Technique 4: The Greedy Algorithm

$$\sum_{j \in I} w_j \le \sum_{\ell=1}^{s} \frac{n_\ell - n_{\ell+1}}{n_\ell} \cdot \text{OPT}$$

$$\le \text{OPT} \sum_{\ell=1}^{s} \left( \frac{1}{n_\ell} + \frac{1}{n_\ell - 1} + \cdots + \frac{1}{n_{\ell+1} + 1} \right)$$

$$= \text{OPT} \sum_{i=1}^{k} \frac{1}{i}$$

$$= H_n \cdot \text{OPT} \le \text{OPT}(\ln n + 1) \ .$$

## Technique 4: The Greedy Algorithm

**A tight example:**
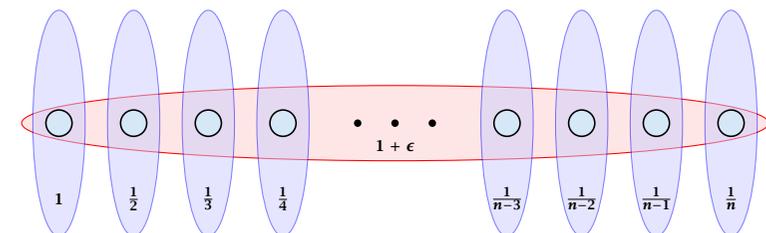
## Technique 5: Randomized Rounding

One round of randomized rounding:
Pick set $S_j$ uniformly at random with probability $1 - x_j$ (for all $j$).

**Version A:** Repeat rounds until you have a cover.

**Version B:** Repeat for $s$ rounds. If you have a cover STOP.
Otherwise, repeat the whole algorithm.

**Probability that $u \in U$ is not covered (in one round):**

$$\Pr[u \text{ not covered in one round}]$$
$$= \prod_{j:u\in S_j} (1 - x_j) \le \prod_{j:u\in S_j} e^{-x_j}$$
$$= e^{-\sum_{j:u\in S_j} x_j} \le e^{-1} .$$

**Probability that $u \in U$ is not covered (after $\ell$ rounds):**

$$\Pr[u \text{ not covered after } \ell \text{ round}] \le \frac{1}{e^\ell} .$$

$$\Pr[\exists u \in U \text{ not covered after } \ell \text{ round}]$$
$$= \Pr[u_1 \text{ not covered} \lor u_2 \text{ not covered} \lor \ldots \lor u_n \text{ not covered}]$$
$$\le \sum_i \Pr[u_i \text{ not covered after } \ell \text{ rounds}] \le ne^{-\ell} .$$

**Lemma 71**
*With high probability $\mathcal{O}(\log n)$ rounds suffice.*

**With high probability:**
For any constant $\alpha$ the number of rounds is at most $\mathcal{O}(\log n)$
with probability at least $1 - n^{-\alpha}$.

**Proof:** We have

$$\Pr[\#\text{rounds} \ge (\alpha + 1) \ln n] \le ne^{-(\alpha+1)\ln n} = n^{-\alpha} .$$

## Expected Cost

- Version A.
  Repeat for $s = (\alpha + 1) \ln n$ rounds. If you don't have a cover simply take for each element $u$ the cheapest set that contains $u$.

  $$E[\text{cost}] \leq (\alpha+1) \ln n \cdot \text{cost}(LP) + (n \cdot \text{OPT}) n^{-\alpha} = \mathcal{O}(\ln n) \cdot \text{OPT}$$

## Expected Cost

- Version B.
  Repeat for $s = (\alpha + 1) \ln n$ rounds. If you don't have a cover simply repeat the whole process.

  $$E[\text{cost}] = \Pr[\text{success}] \cdot E[\text{cost} \mid \text{success}]$$
  $$+ \Pr[\text{no success}] \cdot E[\text{cost} \mid \text{no success}]$$

  This means

  $E[\text{cost} \mid \text{success}]$

  $$= \frac{1}{\Pr[\text{succ.}]} \Big( E[\text{cost}] - \Pr[\text{no success}] \cdot E[\text{cost} \mid \text{no success}] \Big)$$

  $$\leq \frac{1}{\Pr[\text{succ.}]} E[\text{cost}] \leq \frac{1}{1 - n^{-\alpha}} (\alpha + 1) \ln n \cdot \text{cost}(LP)$$

  $$\leq 2(\alpha + 1) \ln n \cdot \text{OPT}$$

  for $n \geq 2$ and $\alpha \geq 1$.

Randomized rounding gives an $\mathcal{O}(\log n)$ approximation. The running time is polynomial with high probability.

### Theorem 72 (without proof)

*There is no approximation algorithm for set cover with approximation guarantee better than $\frac{1}{2} \log n$ unless* NP *has quasi-polynomial time algorithms (algorithms with running time $2^{\text{poly}(\log n)}$).*

## Integrality Gap

The integrality gap of the SetCover LP is $\Omega(\log n)$.

- $n = 2^k - 1$
- Elements are all vectors $\vec{x}$ over $GF[2]$ of length $k$ (excluding zero vector).
- Every vector $\vec{y}$ defines a set as follows

  $$S_{\vec{y}} := \{\vec{x} \mid \vec{x}^T \vec{y} = 1\}$$

- each set contains $2^{k-1}$ vectors; each vector is contained in $2^{k-1}$ sets
- $x_i = \frac{1}{2^{k-1}} = \frac{2}{n+1}$ is fractional solution.

## Integrality Gap

Every collection of $p < k$ sets does not cover all elements.

Hence, we get a gap of $\Omega(\log n)$.

**Techniques:**

- ▶ Deterministic Rounding
- ▶ Rounding of the Dual
- ▶ Primal Dual
- ▶ Greedy
- ▶ Randomized Rounding
- ▶ Local Search
- ▶ Rounding Data + Dynamic Programming

## Scheduling Jobs on Identical Parallel Machines

Given $n$ jobs, where job $j \in \{1, \ldots, n\}$ has processing time $p_j$. Schedule the jobs on $m$ identical parallel machines such that the Makespan (finishing time of the last job) is minimized.

$$
\begin{array}{lrcl}
\min & & & L \\
\text{s.t.} & \forall \text{machines } i \quad \sum_j p_j \cdot x_{j,i} & \leq & L \\
& \forall \text{jobs } j \quad \sum_i x_{j,i} \geq 1 & & \\
& \forall i, j \quad x_{j,i} & \in & \{0, 1\}
\end{array}
$$

Here the variable $x_{j,i}$ is the decision variable that describes whether job $j$ is assigned to machine $i$.

## Lower Bounds on the Solution

Let for a given schedule $C_j$ denote the finishing time of machine $j$, and let $C_{\max}$ be the makespan.

Let $C_{\max}^*$ denote the makespan of an optimal solution.

Clearly

$$
C_{\max}^* \geq \max_j p_j
$$

as the longest job needs to be scheduled somewhere.

## Lower Bounds on the Solution

The average work performed by a machine is $\frac{1}{m}\sum_j p_j$.
Therefore,

$$C_{\max}^* \geq \frac{1}{m}\sum_j p_j$$

## Local Search

A local search algorithm successively makes certain small (cost/profit improving) changes to a solution until it does not find such changes anymore.

It is conceptionally very different from a Greedy algorithm as a feasible solution is always maintained.

Sometimes the running time is difficult to prove.

## Local Search for Scheduling

**Local Search Strategy:** Take the job that finishes last and try to move it to another machine. If there is such a move that reduces the makespan, perform the switch.

REPEAT

## Local Search Analysis

Let $\ell$ be the job that finishes last in the produced schedule.

Let $S_\ell$ be its start time, and let $C_\ell$ be its completion time.

Note that every machine is busy before time $S_\ell$, because otherwise we could move the job $\ell$ and hence our schedule would not be locally optimal.

We can split the total processing time into two intervals one from $0$ to $S_\ell$ the other from $S_\ell$ to $C_\ell$.

The interval $[S_\ell, C_\ell]$ is of length $p_\ell \leq C^*_{\max}$.

During the first interval $[0, S_\ell]$ all processors are busy, and, hence, the total work performed in this interval is

$$m \cdot S_\ell \leq \sum_{j \neq \ell} p_j \ .$$

Hence, the length of the schedule is at most

$$p_\ell + \frac{1}{m} \sum_{j \neq \ell} p_j = (1 - \frac{1}{m}) p_\ell + \frac{1}{m} \sum_j p_j \leq (2 - \frac{1}{m}) C^*_{\max}$$

## A Tight Example

$$p_\ell \approx S_\ell + \frac{S_\ell}{m - 1}$$

$$\frac{\text{ALG}}{\text{OPT}} = \frac{S_\ell + p_\ell}{p_\ell} \approx \frac{2 + \frac{1}{m-1}}{1 + \frac{1}{m-1}} = 2 - \frac{1}{m}$$
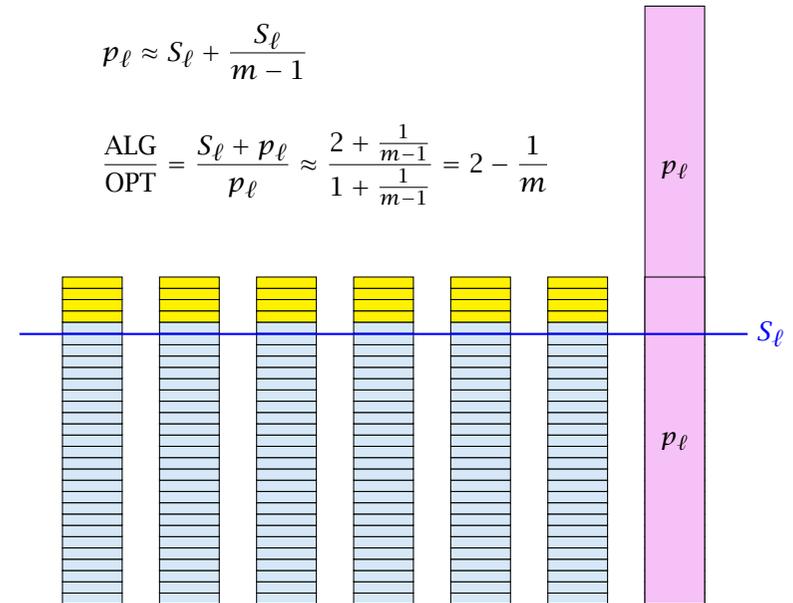


## A Greedy Strategy

**List Scheduling:**
Order all processes in a list. When a machine runs empty assign the next yet unprocessed job to it.

Alternatively:
Consider processes in some order. Assign the $i$-th process to the least loaded machine.

It is easy to see that the result of these greedy strategies fulfill the local optimally condition of our local search algorithm. Hence, these also give 2-approximations.

## A Greedy Strategy

**Lemma 73**
*If we order the list according to non-increasing processing times the approximation guarantee of the list scheduling strategy improves to $4/3$.*

**Proof:**

- Let $p_1 \geq \cdots \geq p_n$ denote the processing times of a set of jobs that form a counter-example.

- Wlog. the last job to finish is $n$ (otw. deleting this job gives another counter-example with fewer jobs).

- If $p_n \leq C^*_{\max}/3$ the previous analysis gives us a schedule length of at most

$$C^*_{\max} + p_n \leq \frac{4}{3} C^*_{\max} \ .$$

  Hence, $p_n > C^*_{\max}/3$.

- This means that all jobs must have a processing time $> C^*_{\max}/3$.

- But then any machine in the optimum schedule can handle at most two jobs.

- For such instances Longest-Processing-Time-First is optimal.

---

When in an optimal solution a machine can have at most 2 jobs the optimal solution looks as follows.

---
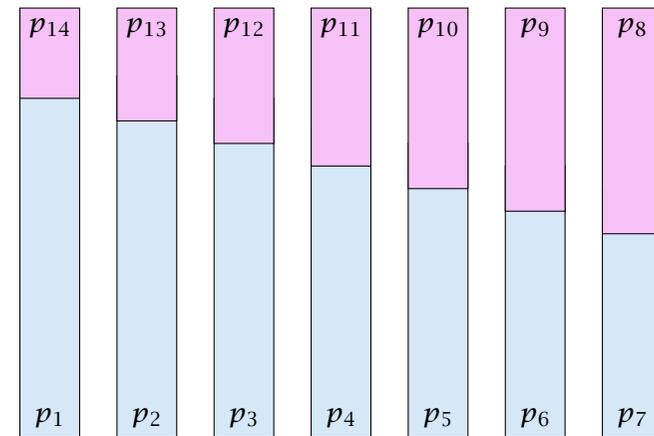
- We can assume that one machine schedules $p_1$ and $p_n$ (the largest and smallest job).

- If not assume wlog. that $p_1$ is scheduled on machine $A$ and $p_n$ on machine $B$.

- Let $p_A$ and $p_B$ be the other job scheduled on $A$ and $B$, respectively.

- $p_1 + p_n \leq p_1 + p_A$ and $p_A + p_B \leq p_1 + p_A$, hence scheduling $p_1$ and $p_n$ on one machine and $p_A$ and $p_B$ on the other, cannot increase the Makespan.

- Repeat the above argument for the remaining machines.

---

## Tight Example

- $2m + 1$ jobs

- 2 jobs with length $2m, 2m - 1, 2m - 2, \ldots, m + 1$ ($2m - 2$ jobs in total)

- 3 jobs of length $m$
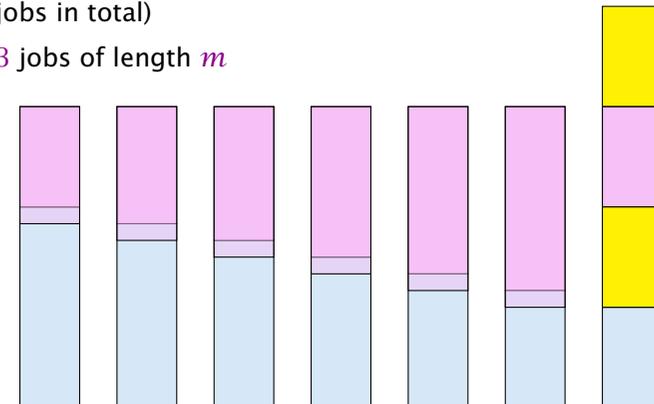
# Traveling Salesman

Given a set of cities ($\{1, \ldots, n\}$) and a symmetric matrix $C = (c_{ij})$, $c_{ij} \geq 0$ that specifies for every pair $(i, j) \in [n] \times [n]$ the cost for travelling from city $i$ to city $j$. Find a permutation $\pi$ of the cities such that the round-trip cost

$$c_{\pi(1)\pi(n)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$$

is minimized.

# Traveling Salesman

### Theorem 74
*There does not exist an $O(2^n)$-approximation algorithm for TSP.*

**Hamiltonian Cycle**:
For a given undirected graph $G = (V, E)$ decide whether there exists a simple cycle that contains all nodes in $G$.

- ▶ Given an instance to HAMPATH we create an instance for TSP.
- ▶ If $(i, j) \notin E$ then set $c_{ij}$ to $n2^n$ otw. set $c_{ij}$ to 1. This instance has polynomial size.
- ▶ There exists a Hamiltonian Path iff there exists a tour with cost $n$. Otw. any tour has cost strictly larger than $2^n$.
- ▶ An $\mathcal{O}(2^n)$-approximation algorithm could decide btw. these cases. Hence, cannot exist unless $P = NP$.

# Metric Traveling Salesman

In the metric version we assume for every triple $i, j, k \in \{1, \ldots, n\}$

$$c_{ij} \leq c_{ij} + c_{jk} \ .$$

It is convenient to view the input as a complete undirected graph $G = (V, E)$, where $c_{ij}$ for an edge $(i, j)$ defines the distance between nodes $i$ and $j$.

# TSP: Lower Bound I

### Lemma 75
*The cost $\mathrm{OPT}_{TSP}(G)$ of an optimum traveling salesman tour is at least as large as the weight $\mathrm{OPT}_{MST}(G)$ of a minimum spanning tree in $G$.*

**Proof:**

- ▶ Take the optimum TSP-tour.
- ▶ Delete one edge.
- ▶ This gives a spanning tree of cost at most $\mathrm{OPT}_{TSP}(G)$.

## TSP: Greedy Algorithm

- ▶ Start with a tour on a subset $S$ containing a single node.
- ▶ Take the node $v$ closest to $S$. Add it $S$ and expand the existing tour on $S$ to include $v$.
- ▶ Repeat until all nodes have been processed.

## TSP: Greedy Algorithm
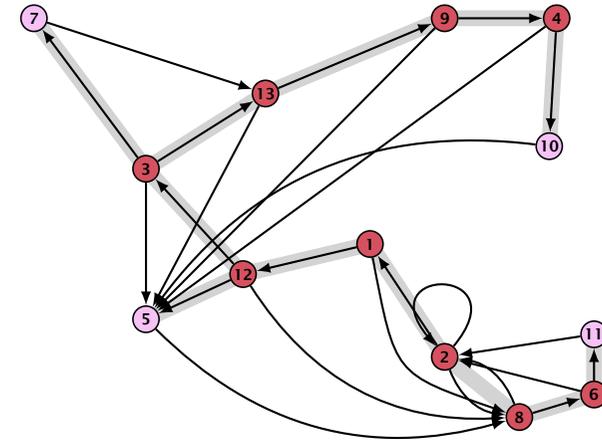


The gray edges form an MST, because exactly these edges are taken in Prims algorithm.

## TSP: Greedy Algorithm

### Lemma 76
*The Greedy algorithm is a $2$-approximation algorithm.*

Let $S_i$ be the set at the start of the $i$-th iteration, and let $v_i$ denote the node added during the iteration.

Further let $s_i \in S_i$ be the node closest to $v_i \in S_i$.

Let $r_i$ denote the successor of $s_i$ in the tour before inserting $v_i$.

We replace the edge $(s_i, r_i)$ in the tour by the two edges $(s_i, v_i)$ and $(v_i, r_i)$.

This increases the cost by

$$c_{s_i,v_i} + c_{v_i,r_i} - c_{s_i,r_i} \leq 2c_{s_i,v_i}$$

## TSP: Greedy Algorithm

The edges $(s_i, v_i)$ considered during the Greedy algorithm are exactly the edges considered during PRIMs MST algorithm.

Hence,

$$\sum_i c_{s_i,v_i} = \mathrm{OPT}_{\mathrm{MST}}(G)$$

which with the previous lower bound gives a $2$-approximation.

## TSP: A different approach

Suppose that we are given an Eulerian graph $G' = (V, E', c')$ of $G = (V, E, c)$ such that for any edge $(i, j) \in E'$ $c'(i, j) \geq c(i, j)$.

Then we can find a TSP-tour of cost at most

$$\sum_{e \in E'} c'(e)$$

▶ Find an Euler tour of $G'$.
▶ Fix a permutation of the cities (i.e., a TSP-tour) by traversing the Euler tour and only note the first occurrence of a city.
▶ The cost of this TSP tour is at most the cost of the Euler tour because of triangle inequality.

This technique is known as short cutting the Euler tour.

## TSP: A different approach
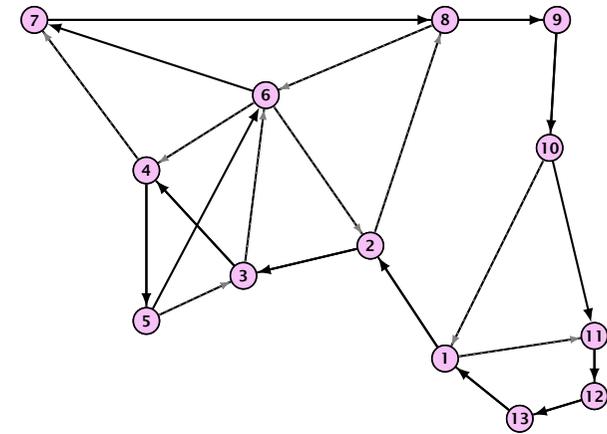
## TSP: A different approach

Consider the following graph:
▶ Compute an MST of $G$.
▶ Duplicate all edges.

This graph is Eulerian, and the total cost of all edges is at most $2 \cdot \mathrm{OPT}_{\mathrm{MST}}(G)$.

Hence, short-cutting gives a tour of cost no more than $2 \cdot \mathrm{OPT}_{\mathrm{MST}}(G)$ which means we have a $2$-approximation.

## TSP: Can we do better?
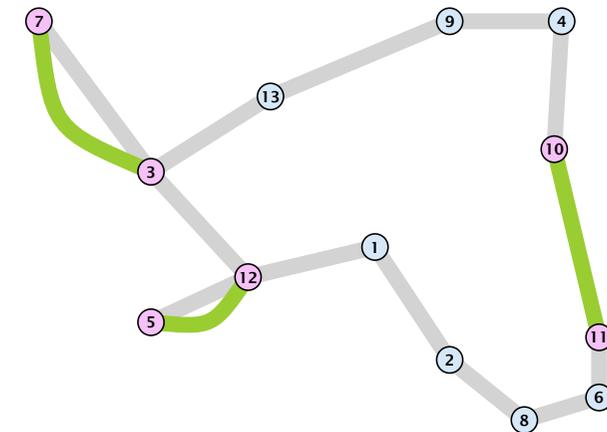
## TSP: Can we do better?

Duplicating all edges in the MST seems to be rather wasteful.

We only need to make the graph Eulerian.

For this we compute a Minimum Weight Matching between odd degree vertices in the MST (note that there are an even number of them).

## TSP: Can we do better?

An optimal tour on the odd-degree vertices has cost at most $\text{OPT}_{\text{TSP}}(G)$.

However, the edges of this tour give rise to two disjoint matchings. One of these matchings must have weight less than $\text{OPT}_{\text{TSP}}(G)/2$.

Adding this matching to the MST gives an Eulerian graph with edge weight at most

$$\text{OPT}_{\text{MST}}(G) + \text{OPT}_{\text{TSP}}(G)/2 \leq \frac{3}{2}\text{OPT}_{\text{TSP}}(G) ,$$

Short cutting gives a $\frac{3}{2}$-approximation for metric TSP.

This is the best that is known.

## Christofides. Tight Example
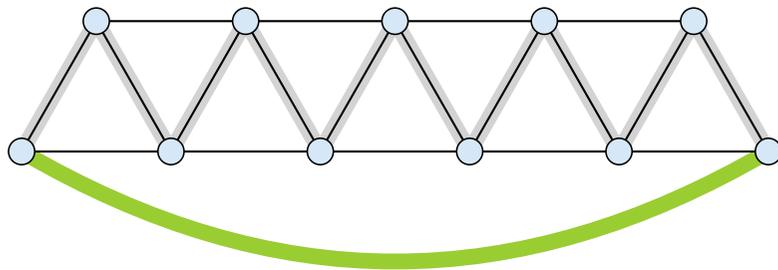


- ▶ optimal tour: $n$ edges.
- ▶ MST: $n - 1$ edges.
- ▶ weight of matching $(n + 1)/2 - 1$
- ▶ MST+matching $\approx 3/2 \cdot n$

## Tree shortcutting. Tight Example
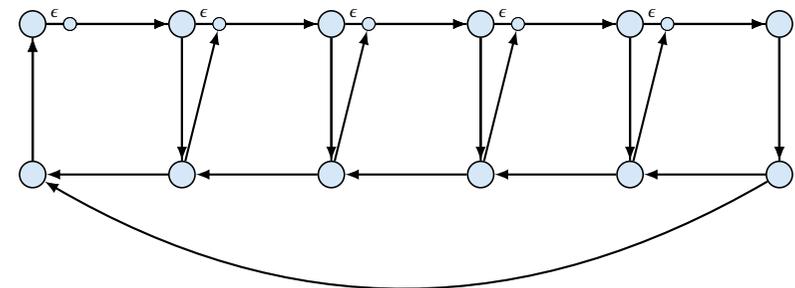


- ▶ edges have Euclidean distance.

# 17 Rounding Data + Dynamic Programming

**Knapsack:**

Given a set of items $\{1,\ldots,n\}$, where the $i$-th item has weight $w_i \in \mathbb{N}$ and profit $p_i \in \mathbb{N}$, and given a threshold $W$. Find a subset $I \subseteq \{1,\ldots,n\}$ of items of total weight at most $W$ such that the profit is maximized (we can assume each $w_i \leq W$).

$$
\begin{array}{lrcl}
\max & \sum_{i=1}^{n} p_i x_i & & \\
\text{s.t.} & \sum_{i=1}^{n} w_i x_i & \leq & W \\
\forall i \in \{1,\ldots,n\} & x_i & \in & \{0,1\}
\end{array}
$$

# 17 Rounding Data + Dynamic Programming

**Algorithm 1** Knapsack
1: $A(1) \leftarrow [(0,0),(p_1,w_1)]$
2: **for** $j \leftarrow 2$ to $n$ **do**
3:    $A(j) \leftarrow A(j-1)$
4:    **for** each $(p,w) \in A(j-1)$ **do**
5:      **if** $w + w_j \leq W$ **then**
6:        add $(p + p_j, w + w_j)$ to $A(j)$
7:      remove dominated pairs from $A(j)$
8: **return** $\max_{(p,w) \in A(n)} p$

The running time is $\mathcal{O}(n \cdot \min\{W,P\})$, where $P = \sum_i p_i$ is the total profit of all items. This is only pseudo-polynomial.

# 17 Rounding Data + Dynamic Programming

**Definition 77**

An algorithm is said to have pseudo-polynomial running time if the running time is polynomial when the numerical part of the input is encoded in unary.

# 17 Rounding Data + Dynamic Programming

- ▶ Let $M$ be the maximum profit of an element.
- ▶ Set $\mu := \epsilon M / n$.
- ▶ Set $p_i' := \lfloor p_i / \mu \rfloor$ for all $i$.
- ▶ Run the dynamic programming algorithm on this revised instance.

Running time is at most

$$
\mathcal{O}(nP') = \mathcal{O}\left(n \sum_i p_i'\right) = \mathcal{O}\left(n \sum_i \left\lfloor \frac{p_i}{\epsilon M / n} \right\rfloor\right) \leq \mathcal{O}\left(\frac{n^3}{\epsilon}\right) .
$$

# 17 Rounding Data + Dynamic Programming

Let $S$ be the set of items returned by the algorithm, and let $O$ be an optimum set of items.

$$
\begin{aligned}
\sum_{i \in S} p_i &\geq \mu \sum_{i \in S} p_i' \\
&\geq \mu \sum_{i \in O} p_i' \\
&\geq \sum_{i \in O} p_i - |O|\mu \\
&\geq \sum_{i \in O} p_i - n\mu \\
&= \sum_{i \in O} p_i - \epsilon M \\
&\geq (1 - \epsilon)\text{OPT} .
\end{aligned}
$$

# Scheduling Revisited

The previous analysis of the scheduling algorithm gave a makespan of

$$
\frac{1}{m} \sum_{j \neq \ell} p_j + p_\ell
$$

where $\ell$ is the last job to complete.

Together with the obervation that if each $p_i \geq \frac{1}{3} C_{\max}^*$ then LPT is optimal this gave a $4/3$-approximation.

# 17.2 Scheduling Revisited

Partition the input into long jobs and short jobs.

A job $j$ is called short if

$$
p_j \leq \frac{1}{km} \sum_i p_i
$$

**Idea:**

1. Find the optimum Makespan for the long jobs by brute force.
2. Then use the list scheduling algorithm for the short jobs, always assigning the next job to the least loaded machine.

We still have the inequality

$$
\frac{1}{m} \sum_{j \neq \ell} p_j + p_\ell
$$

where $\ell$ is the last job (this only requires that all machines are busy before time $S_\ell$).

If $\ell$ is a long job, then the schedule must be optimal, as it consists of an optimal schedule of long jobs plus a schedule for short jobs.

If $\ell$ is a short job its length is at most

$$
p_\ell \leq \sum_j p_j / (mk)
$$

which is at most $C_{\max}^*/k$.

Hence we get a schedule of length at most

$$\left(1 + \frac{1}{k}\right) C_{\max}^*$$

There are at most $km$ long jobs. Hence, the number of possibilities of scheduling these jobs on $m$ machines is at most $m^{km}$, which is constant if $m$ is constant. Hence, it is easy to implement the algorithm in polynomial time.

### Theorem 78
*The above algorithm gives a polynomial time approximation scheme (PTAS) for the problem of scheduling $n$ jobs on $m$ identical machines if $m$ is* constant.

We choose $k = \lceil \frac{1}{\epsilon} \rceil$.

How to get rid of the requirement that $m$ is constant?

We first design an algorithm that works as follows:
On input of $T$ it either finds a schedule of length $(1 + \frac{1}{k})T$ or certifies that no schedule of length at most $T$ exists (assume $T \geq \frac{1}{m} \sum_j p_j$).

We partition the jobs into long jobs and short jobs:

▶ A job is long if its size is larger than $T/k$.
▶ Otw. it is a short job.

▶ We round all long jobs down to multiples of $T/k^2$.
▶ For these rounded sizes we first find an optimal schedule.
▶ If this schedule does not have length at most $T$ we conclude that also the original sizes don't allow such a schedule.
▶ If we have a good schedule we extend it by adding the short jobs according to the LPT rule.

After the first phase the rounded sizes of the long jobs assigned to a machine add up to at most $T$.

There can be at most $k$ (long) jobs assigned to a machine as otw. their rounded sizes would add up to more than $T$ (note that the rounded size of a long job is at least $T/k$).

Since, jobs had been rounded to multiples of $T/k^2$ going from rounded sizes to original sizes gives that the Makespan is at most

$$\left(1 + \frac{1}{k}\right)T \ .$$

During the second phase there always must exist a machine with load at most $T$, since $T$ is larger than the average load. Assigning the current (short) job to such a machine gives that the new load is at most

$$T + \frac{T}{k} \le \left(1 + \frac{1}{k}\right)T \ .$$

---

**Running Time for scheduling large jobs:** There should not be a job with rounded size more than $T$ as otw. the problem becomes trivial.

Hence, any large job has rounded size of $\frac{i}{k^2}T$ for $i \in \{k, \ldots, k^2\}$. Therefore the number of different inputs is at most $n^{k^2}$ (described by a vector of length $k^2$ where, the $i$-th entry describes the number of jobs of size $\frac{i}{k^2}T$). This is polynomial.

The schedule/configuration of a particular machine $x$ can be described by a vector of length $k^2$ where the $i$-th entry describes the number of jobs of rounded size $\frac{i}{k^2}T$ assigned to $x$. There are only $(k+1)^{k^2}$ different vectors.

This means there are a constant number of different machine configurations.

---

Let $\mathrm{OPT}(n_1, \ldots, n_{k^2})$ be the number of machines that are required to schedule input vector $(n_1, \ldots, n_{k^2})$ with Makespan at most $T$.

**If $\mathrm{OPT}(n_1, \ldots, n_{k^2}) \le m$ we can schedule the input.**

We have

$$\mathrm{OPT}(n_1, \ldots, n_{k^2})$$

$$= \begin{cases} 0 & (n_1, \ldots, n_{k^2}) = 0 \\ 1 + \min_{(s_1, \ldots, s_{k^2}) \in C} \mathrm{OPT}(n_1 - s_1, \ldots, n_{k^2} - s_{k^2}) & (n_1, \ldots, n_{k^2}) \gneqq 0 \\ \infty & \text{otw.} \end{cases}$$

where $C$ is the set of all configurations.

Hence, the running time is roughly $(k+1)^{k^2} n^{k^2} \approx (nk)^{k^2}$.

---

We can turn this into a PTAS by choosing $k = \lceil 1/\epsilon \rceil$ and using binary search. This gives a running time that is exponential in $1/\epsilon$.

Can we do better?
Scheduling on identical machines with the goal of minimizing Makespan is a strongly NP-complete problem.

**Theorem 79**
*There is no FPTAS for problems that are strongly NP-hard.*

- Suppose we have an instance with polynomially bounded processing times $p_i \leq q(n)$
- We set $k := \lceil 2nq(n) \rceil \geq 2\,\mathrm{OPT}$
- Then

$$\mathrm{ALG} \leq \left(1 + \frac{1}{k}\right) \mathrm{OPT} \leq \mathrm{OPT} + \frac{1}{2}$$

- But this means that the algorithm computes the optimal solution as the optimum is integral.
- This means we can solve problem instances if processing times are polynomially bounded
- Running time is $\mathcal{O}(\mathrm{poly}(n,k)) = \mathcal{O}(\mathrm{poly}(n))$
- For strongly NP-complete problems this is not possible unless P=NP

## More General

Let $\mathrm{OPT}(n_1, \ldots, n_A)$ be the number of machines that are required to schedule input vector $(n_1, \ldots, n_A)$ with Makespan at most $T$ ($A$: number of different sizes).

If $\mathrm{OPT}(n_1, \ldots, n_A) \leq m$ we can schedule the input.

$$\mathrm{OPT}(n_1, \ldots, n_A)$$
$$= \begin{cases} 0 & (n_1, \ldots, n_A) = 0 \\ 1 + \min_{(s_1, \ldots, s_A) \in C} \mathrm{OPT}(n_1 - s_1, \ldots, n_A - s_A) & (n_1, \ldots, n_A) \gneqq 0 \\ \infty & \text{otw.} \end{cases}$$

where $C$ is the set of all configurations.

$|C| \leq (B+1)^A$, where $B$ is the number of jobs that possibly can fit on the same machine.

The running time is then $O((B+1)^A n^A)$ because the dynamic programming table has just $n^A$ entries.

## Bin Packing

Given $n$ items with sizes $s_1, \ldots, s_n$ where

$$1 > s_1 \geq \cdots \geq s_n > 0 \ .$$

Pack items into a minimum number of bins where each bin can hold items of total size at most 1.

### Theorem 80
*There is no $\rho$-approximation for Bin Packing with $\rho < 3/2$ unless* P = NP.

## Bin Packing

### Proof
- In the partition problem we are given positive integers $b_1, \ldots, b_n$ with $B = \sum_i b_i$ even. Can we partition the integers into two sets $S$ and $T$ s.t.

$$\sum_{i \in S} b_i = \sum_{i \in T} b_i \quad ?$$

- We can solve this problem by setting $s_i := 2b_i/B$ and asking whether we can pack the resulting items into 2 bins or not.
- A $\rho$-approximation algorithm with $\rho < 3/2$ cannot output 3 or more bins when 2 are optimal.
- Hence, such an algorithm can solve Partition.

## Bin Packing

### Definition 81
An asymptotic polynomial-time approximation scheme (APTAS) is a family of algorithms $\{A_\epsilon\}$ along with a constant $c$ such that $A_\epsilon$ returns a solution of value at most $(1 + \epsilon)\mathrm{OPT} + c$ for minimization problems.

- ▶ Note that for Set Cover or for Knapsack it makes no sense to differentiate between the notion of a PTAS or an APTAS because of scaling.
- ▶ However, we will develop an APTAS for Bin Packing.

## Bin Packing

Again we can differentiate between small and large items.

### Lemma 82
*Any packing of items into $\ell$ bins can be extended with items of size at most $\gamma$ s.t. we use only $\max\{\ell, \frac{1}{1-\gamma}\mathrm{SIZE}(I) + 1\}$ bins, where $\mathrm{SIZE}(I) = \sum_i s_i$ is the sum of all item sizes.*

- ▶ If after Greedy we use more than $\ell$ bins, all bins (apart from the last) must be full to at least $1 - \gamma$.
- ▶ Hence, $r(1 - \gamma) \leq \mathrm{SIZE}(I)$ where $r$ is the number of nearly-full bins.
- ▶ This gives the lemma.

Choose $\gamma = \epsilon/2$. Then we either use $\ell$ bins or at most

$$\frac{1}{1 - \epsilon/2} \cdot \mathrm{OPT} + 1 \leq (1 + \epsilon) \cdot \mathrm{OPT} + 1$$

bins.

It remains to find an algorithm for the large items.

## Bin Packing

**Linear Grouping:**
Generate an instance $I'$ (for large items) as follows.

- ▶ Order large items according to size.
- ▶ Let the first $k$ items belong to group 1; the following $k$ items belong to group 2; etc.
- ▶ Delete items in the first group;
- ▶ Round items in the remaining groups to the size of the largest item in the group.

## Linear Grouping

---
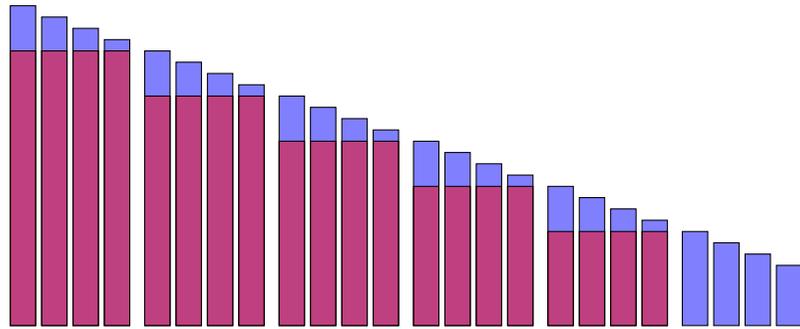
**Lemma 83**
$\text{OPT}(I') \leq \text{OPT}(I) \leq \text{OPT}(I') + k$

**Proof 1:**
- ▶ Any bin packing for $I$ gives a bin packing for $I'$ as follows.
- ▶ Pack the items of group $2$, where in the packing for $I$ the items for group $1$ have been packed;
- ▶ Pack the items of groups $3$, where in the packing for $I$ the items for group $2$ have been packed;
- ▶ …

---

**Lemma 84**
$\text{OPT}(I') \leq \text{OPT}(I) \leq \text{OPT}(I') + k$

**Proof 2:**
- ▶ Any bin packing for $I'$ gives a bin packing for $I$ as follows.
- ▶ Pack the items of group $1$ into $k$ new bins;
- ▶ Pack the items of groups $2$, where in the packing for $I'$ the items for group $2$ have been packed;
- ▶ …

---

Assume that our instance does not contain pieces smaller than $\epsilon/2$. Then $\text{SIZE}(I) \geq \epsilon n/2$.

We set $k = \lfloor \epsilon \text{SIZE}(I) \rfloor$.

Then $n/k \leq n/\lfloor \epsilon^2 n/2 \rfloor \leq 4/\epsilon^2$ (here we used $\lfloor \alpha \rfloor \geq \alpha/2$ for $\alpha \geq 1$).

Hence, after grouping we have a constant number of piece sizes ($4/\epsilon^2$) and at most a constant number ($2/\epsilon$) can fit into any bin.

We can find an optimal packing for such instances by the previous Dynamic Programming approach.

- ▶ cost (for large items) at most

$$\text{OPT}(I') + k \leq \text{OPT}(I) + \epsilon \text{SIZE}(I) \leq (1 + \epsilon)\text{OPT}(I)$$

- ▶ running time $\mathcal{O}((\frac{2}{\epsilon}n)^{4/\epsilon^2})$.

**Can we do better?**

In the following we show how to obtain a solution where the number of bins is only

$$\mathrm{OPT}(I) + \mathcal{O}(\log^2(\mathrm{SIZE}(I))) \ .$$

Note that this is usually better than a guarantee of

$$(1 + \epsilon)\mathrm{OPT}(I) + 1 \ .$$

# Configuration LP

**Change of Notation:**
- Group pieces of identical size.
- Let $s_1$ denote the largest size, and let $b_1$ denote the number of pieces of size $s_1$.
- $s_2$ is second largest size and $b_2$ number of pieces of size $s_2$;
- ...
- $s_m$ smallest size and $b_m$ number of pieces of size $s_m$.

# Configuration LP

A possible packing of a bin can be described by an $m$-tuple $(t_1, \dots, t_m)$, where $t_i$ describes the number of pieces of size $s_i$. Clearly,

$$\sum_i t_i \cdot s_i \le 1 \ .$$

We call a vector that fulfills the above constraint a configuration.

# Configuration LP

Let $N$ be the number of configurations (exponential).

Let $T_1, \dots, T_N$ be the sequence of all possible configurations (a configuration $T_j$ has $T_{ji}$ pieces of size $s_i$).

$$
\begin{array}{llccc}
\min & & \sum_{j=1}^{N} x_j & & \\
\text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \ge & b_i \\
& \forall j \in \{1, \dots, N\} & x_j & \ge & 0 \\
& \forall j \in \{1, \dots, N\} & x_j & \text{integral} &
\end{array}
$$

**How to solve this LP?**

later...

We can assume that each item has size at least $1/\text{SIZE}(I)$.

# Harmonic Grouping

- ▶ Sort items according to size (monotonically decreasing).
- ▶ Process items in this order; close the current group if size of items in the group is at least $2$ (or larger). Then open new group.
- ▶ I.e., $G_1$ is the smallest cardinality set of largest items s.t. total size sums up to at least $2$. Similarly, for $G_2, \ldots, G_{r-1}$.
- ▶ Only the size of items in the last group $G_r$ may sum up to less than $2$.

# Harmonic Grouping

From the grouping we obtain instance $I'$ as follows:

- ▶ Round all items in a group to the size of the largest group member.
- ▶ Delete all items from group $G_1$ and $G_r$.
- ▶ For groups $G_2, \ldots, G_{r-1}$ delete $n_i - n_{i-1}$ items.
- ▶ Observe that $n_i \geq n_{i-1}$.

## Lemma 85
*The number of different sizes in $I'$ is at most $\text{SIZE}(I)/2$.*

▶ Each group that survives (recall that $G_1$ and $G_r$ are deleted) has total size at least $2$.
▶ Hence, the number of surviving groups is at most $\text{SIZE}(I)/2$.
▶ All items in a group have the same size in $I'$.

## Lemma 86
*The total size of deleted items is at most $\mathcal{O}(\log(\text{SIZE}(I)))$.*

▶ The total size of items in $G_1$ and $G_r$ is at most $6$ as a group has total size at most $3$.
▶ Consider a group $G_i$ that has strictly more items than $G_{i-1}$.
▶ It discards $n_i - n_{i-1}$ pieces of total size at most

$$3\frac{n_i - n_{i-1}}{n_i} \le \sum_{j=n_{i-1}+1}^{n_i} \frac{3}{j}$$

since the smallest piece has size at most $3/n_i$.

▶ Summing over all $i$ that have $n_i > n_{i-1}$ gives a bound of at most

$$\sum_{j=1}^{n_{r-1}} \frac{3}{j} \le \mathcal{O}(\log(\text{SIZE}(I))) \ .$$

(note that $n_r \le \text{SIZE}(I)$ since we assume that the size of each item is at least $1/\text{SIZE}(I)$).

---

**Algorithm 1** BinPack
1: **if** $\text{SIZE}(I) < 10$ **then**
2:     pack remaining items greedily
3: Apply harmonic grouping to create instance $I'$; pack discarded items in at most $\mathcal{O}(\log(\text{SIZE}(I)))$ bins.
4: Let $x$ be optimal solution to configuration LP
5: Pack $\lfloor x_j \rfloor$ bins in configuration $T_j$ for all $j$; call the packed instance $I_1$.
6: Let $I_2$ be remaining pieces from $I'$
7: Pack $I_2$ via BinPack($I_2$)

## Analysis

$$\text{OPT}_{\text{LP}}(I_1) + \text{OPT}_{\text{LP}}(I_2) \le \text{OPT}_{\text{LP}}(I') \le \text{OPT}_{\text{LP}}(I)$$

**Proof:**

▶ Each piece surviving in $I'$ can be mapped to a piece in $I$ of no lesser size. Hence, $\text{OPT}_{\text{LP}}(I') \le \text{OPT}_{\text{LP}}(I)$
▶ $\lfloor x_j \rfloor$ is feasible solution for $I_1$ (even integral).
▶ $x_j - \lfloor x_j \rfloor$ is feasible solution for $I_2$.

## Analysis

Each level of the recursion partitions pieces into three types

1. Pieces discarded at this level.
2. Pieces scheduled because they are in $I_1$.
3. Pieces in $I_2$ are handed down to the next level.

Pieces of type 2 summed over all recursion levels are packed into at most $\text{OPT}_{\text{LP}}$ many bins.

Pieces of type 1 are packed into at most

$$\mathcal{O}(\log(\text{SIZE}(I))) \cdot L$$

many bins where $L$ is the number of recursion levels.

## Analysis

We can show that $\text{SIZE}(I_2) \le \text{SIZE}(I)/2$. Hence, the number of recursion levels is only $\mathcal{O}(\log(\text{SIZE}(I_{\text{original}})))$ in total.

▸ The number of non-zero entries in the solution to the configuration LP for $I'$ is at most the number of constraints, which is the number of different sizes ($\le \text{SIZE}(I)/2$).

▸ The total size of items in $I_2$ can be at most $\sum_{j=1}^{N} x_j - \lfloor x_j \rfloor$ which is at most the number of non-zero entries in the solution to the configuration LP.

## How to solve the LP?

Let $T_1, \ldots, T_N$ be the sequence of all possible configurations (a configuration $T_j$ has $T_{ji}$ pieces of size $s_i$).
In total we have $b_i$ pieces of size $s_i$.

**Primal**

$$
\begin{array}{lll}
\min & \sum_{j=1}^{N} x_j & \\
\text{s.t.} & \forall i \in \{1 \ldots m\} \quad \sum_{j=1}^{N} T_{ji} x_j & \ge \quad b_i \\
& \forall j \in \{1, \ldots, N\} \quad x_j & \ge \quad 0
\end{array}
$$

**Dual**

$$
\begin{array}{lll}
\max & \sum_{i=1}^{m} y_i b_i & \\
\text{s.t.} & \forall j \in \{1, \ldots, N\} \quad \sum_{i=1}^{m} T_{ji} y_i & \le \quad 1 \\
& \forall i \in \{1, \ldots, m\} \quad y_i & \ge \quad 0
\end{array}
$$

## Separation Oracle

Suppose that I am given variable assignment $y$ for the dual.

**How do I find a violated constraint?**

I have to find a configuration $T_j = (T_{j1}, \ldots, T_{jm})$ that

▸ is feasible, i.e.,

$$\sum_{i=1}^{m} T_{ji} \cdot s_i \le 1 \ ,$$

▸ and has a large profit

$$\sum_{i=1}^{m} T_{ji} y_i > 1$$

But this is the Knapsack problem.

## Separation Oracle

We have FPTAS for Knapsack. This means if a constraint is violated with $1 + \epsilon' = 1 + \frac{\epsilon}{1-\epsilon}$ we find it, since we can obtain at least $(1 - \epsilon)$ of the optimal profit.

The solution we get is feasible for:

**Dual′**

$$
\begin{array}{rrcl}
\max & \sum_{i=1}^{m} y_i b_i & & \\
\text{s.t.} \quad \forall j \in \{1,\dots,N\} & \sum_{i=1}^{m} T_{ji} y_i & \leq & 1 + \epsilon' \\
\forall i \in \{1,\dots,m\} & y_i & \geq & 0
\end{array}
$$

**Primal′**

$$
\begin{array}{rrcl}
\min & (1 + \epsilon') \sum_{j=1}^{N} x_j & & \\
\text{s.t.} \quad \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\
\forall j \in \{1,\dots,N\} & x_j & \geq & 0
\end{array}
$$

## Separation Oracle

If the value of the computed dual solution (which may be infeasible) is $z$ then

$$\text{OPT} \leq z \leq (1 + \epsilon')\text{OPT}$$

**How do we get good primal solution (not just the value)?**

- ▶ The constraints used when computing $z$ certify that the solution is feasible for $\text{DUAL}'$.
- ▶ Suppose that we drop all unused constraints in $\text{DUAL}$. We will compute the same solution feasible for $\text{DUAL}'$.
- ▶ Let $\text{DUAL}''$ be $\text{DUAL}$ without unused constraints.
- ▶ The dual to $\text{DUAL}''$ is $\text{PRIMAL}$ where we ignore variables for which the corresponding dual constraint has not been used.
- ▶ The optimum value for $\text{PRIMAL}''$ is at most $(1 + \epsilon')\text{OPT}$.
- ▶ We can compute the corresponding solution in polytime.

This gives that overall we need at most

$$(1 + \epsilon')\text{OPT}_{\text{LP}}(I) + \mathcal{O}(\log^2(\text{SIZE}(I)))$$

bins.

We can choose $\epsilon' = \frac{1}{\text{OPT}}$ as $\text{OPT} \leq \#\text{items}$ and since we have a fully polynomial time approximation scheme (FPTAS) for knapsack.

### Lemma 87 (Chernoff Bounds)

*Let $X_1,\dots,X_n$ be $n$ independent 0-1 random variables, not necessarily identically distributed. Then for $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$, $L \leq \mu \leq U$, and $\delta > 0$*

$$\Pr[X \geq (1 + \delta)U] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^U ,$$

*and*

$$\Pr[X \leq (1 - \delta)L] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^L ,$$

### Lemma 88

For $0 \leq \delta \leq 1$ we have that

$$\left( \frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{U} \leq e^{-U\delta^2/3}$$

and

$$\left( \frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{L} \leq e^{-L\delta^2/2}$$

## Proof of Chernoff Bounds

**Markovs Inequality**:

Let $X$ be random variable taking non-negative values.
Then

$$\Pr[X \geq a] \leq \mathrm{E}[X]/a$$

**Trivial!**

## Proof of Chernoff Bounds

**Hence:**

$$\Pr[X \geq (1+\delta)U] \leq \frac{\mathrm{E}[X]}{(1+\delta)U} \approx \frac{1}{1+\delta}$$

**That's awfully weak :(**

## Proof of Chernoff Bounds

Set $p_i = \Pr[X_i = 1]$. Assume $p_i > 0$ for all $i$.

**Cool Trick:**

$$\Pr[X \geq (1+\delta)U] = \Pr[e^{tX} \geq e^{t(1+\delta)U}]$$

Now, we apply Markov:

$$\Pr[e^{tX} \geq e^{t(1+\delta)U}] \leq \frac{\mathrm{E}[e^{tX}]}{e^{t(1+\delta)U}} \ .$$

**This may be a lot better (!?)**

## Proof of Chernoff Bounds

$$\mathrm{E}\left[e^{tX}\right] = \mathrm{E}\left[e^{t\sum_i X_i}\right] = \mathrm{E}\left[\prod_i e^{tX_i}\right] = \prod_i \mathrm{E}\left[e^{tX_i}\right]$$

$$\mathrm{E}\left[e^{tX_i}\right] = (1 - p_i) + p_i e^t = 1 + p_i(e^t - 1) \leq e^{p_i(e^t - 1)}$$

$$\prod_i \mathrm{E}\left[e^{tX_i}\right] \leq \prod_i e^{p_i(e^t - 1)} = e^{\sum p_i(e^t - 1)} = e^{(e^t - 1)U}$$

---

Now, we apply Markov:

$$\Pr[X \geq (1 + \delta)U] = \Pr[e^{tX} \geq e^{t(1+\delta)U}]$$

$$\leq \frac{\mathrm{E}[e^{tX}]}{e^{t(1+\delta)U}} \leq \frac{e^{(e^t - 1)U}}{e^{t(1+\delta)U}} \leq \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^U$$

We choose $t = \ln(1 + \delta)$.

---

**Lemma 89**

*For $0 \leq \delta \leq 1$ we have that*

$$\left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^U \leq e^{-U\delta^2/3}$$

*and*

$$\left(\frac{e^{-\delta}}{(1 - \delta)^{1-\delta}}\right)^L \leq e^{-L\delta^2/2}$$

---

Show:

$$\left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^U \leq e^{-U\delta^2/3}$$

Take logarithms:

$$U(\delta - (1 + \delta)\ln(1 + \delta)) \leq -U\delta^2/3$$

True for $\delta = 0$. Divide by $U$ and take derivatives:

$$-\ln(1 + \delta) \leq -2\delta/3$$

**Reason:**
As long as derivative of left side is smaller than derivative of right side the inequality holds.

$$f(\delta) := -\ln(1+\delta) + 2\delta/3 \leq 0$$

A convex function ($f''(\delta) \geq 0$) on an interval takes maximum at the boundaries.

$$f'(\delta) = -\frac{1}{1+\delta} + 2/3 \qquad f''(\delta) = \frac{1}{(1+\delta)^2}$$

$f(0) = 0$ and $f(1) = -\ln(2) + 2/3 < 0$

For $\delta \geq 1$ we show

$$\left( \frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^U \leq e^{-U\delta/3}$$

Take logarithms:

$$U(\delta - (1+\delta)\ln(1+\delta)) \leq -U\delta/3$$

True for $\delta = 0$. Divide by $U$ and take derivatives:

$$-\ln(1+\delta) \leq -1/3 \iff \ln(1+\delta) \geq 1/3 \quad \text{(true)}$$

**Reason:**
As long as derivative of left side is smaller than derivative of right side the inequality holds.

Show:

$$\left( \frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^L \leq e^{-L\delta^2/2}$$

Take logarithms:

$$L(-\delta - (1-\delta)\ln(1-\delta)) \leq -L\delta^2/2$$

True for $\delta = 0$. Divide by $L$ and take derivatives:

$$\ln(1-\delta) \leq -\delta$$

**Reason:**
As long as derivative of left side is smaller than derivative of right side the inequality holds.

$$\ln(1-\delta) \leq -\delta$$

True for $\delta = 0$. Take derivatives:

$$-\frac{1}{1-\delta} \leq -1$$

This holds for $0 \leq \delta < 1$.

# Integer Multicommodity Flows

- Given $s_i$-$t_i$ pairs in a graph.
- Connect each pair by a path such that not too many path use any given edge.

$$
\begin{array}{rlrcl}
\min & & W & & \\
\text{s.t.} & \forall i & \sum_{p \in \mathcal{P}_i} x_p & = & 1 \\
& & \sum_{p: e \in p} x_p & \leq & W \\
& & x_p & \in & \{0, 1\}
\end{array}
$$

# Integer Multicommodity Flows

**Randomized Rounding:**
For each $i$ choose one path from the set $\mathcal{P}_i$ at random according to the probability distribution given by the Linear Programming solution.

# Integer Multicommodity Flows

**Theorem 90**
*If $W^* \geq c \ln n$ for some constant $c$, then with probability at least $n^{-c/3}$ the total number of paths using any edge is at most $W^* + \sqrt{cW^* \ln n}$.*

**Theorem 91**
*With probability at least $n^{-c/3}$ the total number of paths using any edge is at most $W^* + c \ln n$.*

# Integer Multicommodity Flows

Let $X_e^i$ be a random variable that indicates whether the path for $s_i$-$t_i$ uses edge $e$.

Then the number of paths using edge $e$ is $Y_e = \sum_i X_e^i$.

$$
E[Y_e] = \sum_i \sum_{p \in \mathcal{P}_i : e \in p} x_p^* = \sum_{p : e \in P} x_p^* \leq W^*
$$

## Integer Multicommodity Flows

Choose $\delta = \sqrt{(c \ln n)/W^*}$.

Then
$$\Pr[Y_e \geq (1+\delta)W^*] < e^{-W^*\delta^2/3} = \frac{1}{n^{c/3}}$$

## 19 MAXSAT

**Problem definition:**

- ▸ $n$ Boolean variables
- ▸ $m$ clauses $C_1, \ldots, C_m$. For example

$$C_7 = x_3 \vee \bar{x}_5 \vee \bar{x}_9$$

- ▸ Non-negative weight $w_j$ for each clause $C_j$.
- ▸ Find an assignment of true/false to the variables sucht that the total weight of clauses that are satisfied is maximum.

## 19 MAXSAT

**Terminology:**

- ▸ A variable $x_i$ and its negation $\bar{x}_i$ are called literals.
- ▸ Hence, each clause consists of a set of literals (i.e., no duplications: $x_i \vee x_i \vee \bar{x}_j$ is **not** a clause).
- ▸ We assume a clause does not contain $x_i$ and $\bar{x}_i$ for any $i$.
- ▸ $x_i$ is called a positive literal while the negation $\bar{x}_i$ is called a negative literal.
- ▸ For a given clause $C_j$ the number of its literals is called its length or size and denoted with $\ell_j$.
- ▸ Clauses of length one are called unit clauses.

## MAXSAT: Flipping Coins

Set each $x_i$ independently to true with probability $\frac{1}{2}$ (and, hence, to false with probability $\frac{1}{2}$, as well).

Define random variable $X_j$ with

$$X_j = \begin{cases} 1 & \text{if } C_j \text{ satisfied} \\ 0 & \text{otw.} \end{cases}$$

Then the total weight $W$ of satisfied clauses is given by

$$W = \sum_j w_j X_j$$

$$
\begin{aligned}
E[W] &= \sum_j w_j E[X_j] \\
&= \sum_j w_j \Pr[C_j \text{ is satisified}] \\
&= \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\
&\geq \frac{1}{2} \sum_j w_j \\
&\geq \frac{1}{2} \text{OPT}
\end{aligned}
$$

## MAXSAT: LP formulation

- ▶ Let for a clause $C_j$, $P_j$ be the set of positive literals and $N_j$ the set of negative literals.

$$C_j = \bigvee_{j \in P_j} x_i \vee \bigvee_{j \in N_j} \bar{x}_i$$

$$
\begin{array}{llrcl}
\max & & \sum_j w_j z_j & & \\
\text{s.t.} & \forall j & \sum_{i \in P_j} y_i + \sum_{i \in N_j}(1 - y_i) & \geq & z_j \\
& \forall i & y_i & \in & \{0,1\} \\
& \forall j & z_j & \leq & 1
\end{array}
$$

## MAXSAT: Randomized Rounding

Set each $x_i$ independently to true with probability $y_i$ (and, hence, to false with probability $(1 - y_i)$).

**Lemma 92 (Geometric Mean ≤ Arithmetic Mean)**

*For any nonnegative $a_1, \ldots, a_k$*

$$\left( \prod_{i=1}^{k} a_i \right)^{1/k} \leq \frac{1}{k} \sum_{i=1}^{k} a_i$$

**Definition 93**

A function $f$ on an interval $I$ is concave if for any two points $s$ and $r$ from $I$ and any $\lambda \in [0,1]$ we have

$$f(\lambda s + (1 - \lambda)r) \geq \lambda f(s) + (1 - \lambda)f(r)$$

**Lemma 94**

*Let $f$ be a concave function on the interval $[0,1]$, with $f(0) = a$ and $f(1) = a + b$. Then*

$$\begin{aligned}
f(\lambda) = f((1 - \lambda)0 + \lambda 1) \\
\geq (1 - \lambda)f(0) + \lambda f(1) \\
= a + \lambda b
\end{aligned}$$

*for $\lambda \in [0,1]$.*

$$\begin{aligned}
\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j}(1 - y_i) \prod_{i \in N_j} y_i \\
&\leq \left[ \frac{1}{\ell_j} \left( \sum_{i \in P_j}(1 - y_i) + \sum_{i \in N_j} y_i \right) \right]^{\ell_j} \\
&= \left[ 1 - \frac{1}{\ell_j} \left( \sum_{i \in P_j} y_i + \sum_{i \in N_j}(1 - y_i) \right) \right]^{\ell_j} \\
&\leq \left( 1 - \frac{z_j}{\ell_j} \right)^{\ell_j} .
\end{aligned}$$

The function $f(z) = 1 - (1 - \frac{z}{\ell})^{\ell}$ is concave. Hence,

$$\begin{aligned}
\Pr[C_j \text{ satisfied}] &\geq 1 - \left( 1 - \frac{z_j}{\ell_j} \right)^{\ell_j} \\
&\geq \left[ 1 - \left( 1 - \frac{1}{\ell_j} \right)^{\ell_j} \right] \cdot z_j .
\end{aligned}$$

$f''(z) = -\frac{\ell - 1}{\ell} \left[ 1 - \frac{z}{\ell} \right]^{\ell - 2} \leq 0$ for $z \in [0,1]$. Therefore, $f$ is concave.

## Slide 417

$$E[W] = \sum_j w_j \Pr[C_j \text{ is satisfied}]$$

$$\geq \sum_j w_j z_j \left[ 1 - \left( 1 - \frac{1}{\ell_j} \right)^{\ell_j} \right]$$

$$\geq \left( 1 - \frac{1}{e} \right) \text{OPT} .$$

## Slide 418

# MAXSAT: The better of two

**Theorem 95**

*Choosing the better of the two solutions given by randomized rounding and coin flipping yields a $\frac{3}{4}$-approximation.*

## Slide 419

Let $W_1$ be the value of randomized rounding and $W_2$ the value obtained by coin flipping.

$$E[\max\{W_1, W_2\}]$$

$$\geq E[\tfrac{1}{2} W_1 + \tfrac{1}{2} W_2]$$

$$\geq \frac{1}{2} \sum_j w_j z_j \left[ 1 - \left( 1 - \frac{1}{\ell_j} \right)^{\ell_j} \right] + \frac{1}{2} \sum_j w_j \left( 1 - \left( \frac{1}{2} \right)^{\ell_j} \right)$$

$$\geq \sum_j w_j z_j \underbrace{\left[ \frac{1}{2} \left( 1 - \left( 1 - \frac{1}{\ell_j} \right)^{\ell_j} \right) + \frac{1}{2} \left( 1 - \left( \frac{1}{2} \right)^{\ell_j} \right) \right]}_{\geq \frac{3}{4} \text{for all integers}}$$

$$\geq \frac{3}{4} \text{OPT}$$

## Slide 420

## MAXSAT: Nonlinear Randomized Rounding

So far we used linear randomized rounding, i.e., the probability that a variable is set to $1$/true was exactly the value of the corresponding variable in the linear program.

We could define a function $f : [0,1] \to [0,1]$ and set $x_i$ to true with probability $f(y_i)$.

## MAXSAT: Nonlinear Randomized Rounding

Let $f : [0,1] \to [0,1]$ be a function with

$$1 - 4^{-x} \le f(x) \le 4^{x-1}$$

**Theorem 96**
*Rounding the LP-solution with a function $f$ of the above form gives a $\frac{3}{4}$-approximation.*

## MAXSAT: Nonlinear Randomized Rounding

$$\Pr[C_j \text{ not satisfied}] = \prod_{i \in P_j} (1 - f(y_i)) \prod_{i \in N_j} f(y_i)$$

$$\le \prod_{i \in P_j} 4^{-y_i} \prod_{i \in N_j} 4^{y_i - 1}$$

$$= 4^{-\left(\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i)\right)}$$

$$\le 4^{-z_j}$$

The function $g(z) = 1 - 4^{-z}$ is concave on $[0,1]$. Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

Therefore,

$$E[W] = \sum_j w_j \Pr[C_j \text{ satisfied}] \geq \frac{3}{4}\sum_j w_j z_j \geq \frac{3}{4}\text{OPT}$$

---

## Can we do better?

Not if we compare ourselves to the value of an optimum LP-solution.

### Definition 97 (Integrality Gap)
The integrality gap for an ILP is the worst-case ratio over all instances of the problem of the value of an optimal IP-solution to the value of an optimal solution to its linear programming relaxation.

Note that the integrality is less than one for maximization problems and larger than one for minimization problems (of course, equality is possible).

Note that an integrality gap only holds for one specific ILP formulation.

---

### Lemma 98
*Our ILP-formulation for the MAXSAT problem has integrality gap at most $\frac{3}{4}$.*

$$
\begin{array}{llrcl}
\max & & \sum_j w_j z_j & & \\
\text{s.t.} & \forall j & \sum_{i \in P_j} y_i + \sum_{i \in N_j}(1 - y_i) & \geq & z_j \\
& \forall i & y_i & \in & \{0,1\} \\
& \forall j & z_j & \leq & 1
\end{array}
$$

Consider: $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

- any solution can satisfy at most 3 clauses
- we can set $y_1 = y_2 = 1/2$ in the LP; this allows to set $z_1 = z_2 = z_3 = z_4 = 1$
- hence, the LP has value 4.

---

## Repetition: Primal Dual for Set Cover

**Primal Relaxation:**

$$
\begin{array}{llrcl}
\min & & \sum_{i=1}^{k} w_i x_i & & \\
\text{s.t.} & \forall u \in U & \sum_{i:u \in S_i} x_i & \geq & 1 \\
& \forall i \in \{1,\dots,k\} & x_i & \geq & 0
\end{array}
$$

**Dual Formulation:**

$$
\begin{array}{llrcl}
\max & & \sum_{u \in U} y_u & & \\
\text{s.t.} & \forall i \in \{1,\dots,k\} & \sum_{u:u \in S_i} y_u & \leq & w_i \\
& & y_u & \geq & 0
\end{array}
$$

## Repetition: Primal Dual for Set Cover

**Algorithm:**

- ▶ Start with $y = 0$ (feasible dual solution).

  Start with $x = 0$ (integral primal solution that may be infeasible).

- ▶ While $x$ not feasible
  - ▶ Identify an element $e$ that is not covered in current primal integral solution.
  - ▶ Increase dual variable $y_e$ until a dual constraint becomes tight (maybe increase by 0!).
  - ▶ If this is the constraint for set $S_j$ set $x_j = 1$ (add this set to your solution).

## Repetition: Primal Dual for Set Cover

**Analysis:**

- ▶ For every set $S_j$ with $x_j = 1$ we have

$$\sum_{e \in S_j} y_e = w_j$$

- ▶ Hence our cost is

$$\sum_j w_j = \sum_j \sum_{e \in S_j} y_e = \sum_e |\{j : e \in S_j\}| \cdot y_e \le f \cdot \sum_e y_e \le f \cdot \text{OPT}$$

Note that the constructed pair of primal and dual solution fulfills primal slackness conditions.

This means

$$x_j > 0 \Rightarrow \sum_{e \in S_j} y_e = w_j$$

If we would also fulfill dual slackness conditions

$$y_e > 0 \Rightarrow \sum_{j : e \in S_j} x_j = 1$$

then the solution would be optimal!!!

We don't fulfill these constraint but we fulfill an approximate version:

$$y_e > 0 \Rightarrow 1 \le \sum_{j : e \in S_j} x_j \le f$$

This is sufficient to show that the solution is an $f$-approximation.

Suppose we have a primal/dual pair

$$\begin{array}{llrcl} \min & & \sum_j c_j x_j & & \\ \text{s.t.} & \forall i & \sum_{j:} a_{ij} x_j & \geq & b_i \\ & \forall j & x_j & \geq & 0 \end{array}$$
$$\begin{array}{llrcl} \max & & \sum_i b_i y_i & & \\ \text{s.t.} & \forall j & \sum_i a_{ij} y_i & \leq & c_j \\ & \forall i & y_i & \geq & 0 \end{array}$$

and solutions that fulfill approximate slackness conditions:

$$x_j > 0 \Rightarrow \sum_i a_{ij} y_i \geq \frac{1}{\alpha} c_j$$

$$y_i > 0 \Rightarrow \sum_j a_{ij} x_j \leq \beta b_i$$

---

Then



right hand side of $j$-th dual constraint

$$\sum_j \boxed{c_j} x_j \leq \alpha \sum_j \left( \sum_i a_{ij} y_i \right) x_j$$

primal cost

$$= \alpha \sum_i \left( \sum_j a_{ij} x_j \right) y_i$$

$$\leq \alpha \beta \cdot \boxed{\sum_i b_i y_i}$$

dual objective

---

# Feedback Vertex Set for Undirected Graphs

▶ Given a graph $G = (V, E)$ and non-negative weights $w_v \geq 0$ for vertex $v \in V$.

▶ Choose a minimum cost subset of vertices s.t. every cycle contains at least one vertex.

---

We can encode this as an instance of Set Cover

▶ Each vertex can be viewed as a set that contains some cycles.

▶ However, this encoding gives a Set Cover instance of non-polynomial size.

▶ The $O(\log n)$-approximation for Set Cover does not help us to get a good solution.

Let $C$ denote the set of all cycles (where a cycle is identified by its set of vertices)

**Primal Relaxation:**

$$
\begin{array}{rrll}
\min & \sum_v w_v x_v & & \\
\text{s.t.} \quad \forall C \in C & \sum_{v \in C} x_v & \geq & 1 \\
\forall v & x_v & \geq & 0
\end{array}
$$

**Dual Formulation:**

$$
\begin{array}{rrll}
\max & \sum_{C \in C} y_C & & \\
\text{s.t.} \quad \forall v \in V & \sum_{C:v \in C} y_C & \leq & w_v \\
\forall C & y_C & \geq & 0
\end{array}
$$

---

If we perform the previous dual technique for Set Cover we get the following:

- Start with $x = 0$ and $y = 0$
- While there is a cycle $C$ that is not covered (does not contain a chosen vertex).
  - Increase $y_C$ until dual constraint for some vertex $v$ becomes tight.
  - set $x_v = 1$.

---

Then

$$
\begin{aligned}
\sum_v w_v x_v &= \sum_v \sum_{C:v \in C} y_C x_v \\
&= \sum_{v \in S} \sum_{C:v \in C} y_C \\
&= \sum_C |S \cap C| \cdot y_C
\end{aligned}
$$

where $S$ is the set of vertices we choose.

If every cycle is short we get a good approximation ratio, but this is unrealistic.

---

**Algorithm 1** FeedbackVertexSet

1: $y \leftarrow 0$
2: $x \leftarrow 0$
3: **while** exists cycle $C$ in $G$ **do**
4:     increase $y_C$ until there is $v \in C$ s.t. $\sum_{C:v \in C} y_C = w_v$
5:     $x_v = 1$
6:     remove $v$ from $G$
7:     repeatedly remove vertices of degree 1 from $G$

**Idea:**

Always choose a short cycle that is not covered. If we always find a cycle of length at most $\alpha$ we get an $\alpha$-approximation.

**Observation:**

For any path $P$ of vertices of degree $2$ in $G$ the algorithm chooses at most one vertex from $P$.

---

**Observation:**

If we always choose a cycle for which the number of vertices of degree at least 3 is at most $\alpha$ we get a $2\alpha$-approximation.

**Theorem 99**

*In any graph with no vertices of degree 1, there always exists a cycle that has at most $\mathcal{O}(\log n)$ vertices of degree 3 or more. We can find such a cycle in linear time.*

This means we have

$$y_C > 0 \Rightarrow |S \cap C| \le \mathcal{O}(\log n) \ .$$

---

# Primal Dual for Shortest Path

Given a graph $G = (V, E)$ with two nodes $s, t \in V$ and edge-weights $c : E \to \mathbb{R}^+$ find a shortest path between $s$ and $t$ w.r.t. edge-weights $c$.

$$
\begin{array}{lrcl}
\min & \sum_e c(e)x_e & & \\
\text{s.t.} \quad \forall S \in S & \sum_{e:\delta(S)} x_e & \ge & 1 \\
\forall e \in E & x_e & \in & \{0,1\}
\end{array}
$$

Here $\delta(S)$ denotes the set of edges with exactly one end-point in $S$, and $S = \{S \subseteq V : s \in S, t \notin S\}$.

---

# Primal Dual for Shortest Path

**The Dual:**

$$
\begin{array}{lrcl}
\max & \sum_S y_S & & \\
\text{s.t.} \quad \forall e \in E & \sum_{S:e\in\delta(S)} y_S & \le & c(e) \\
\forall S \in S & y_S & \ge & 0
\end{array}
$$

Here $\delta(S)$ denotes the set of edges with exactly one end-point in $S$, and $S = \{S \subseteq V : s \in S, t \notin S\}$.

## Primal Dual for Shortest Path

We can interpret the value $y_S$ as the width of a moat surounding the set $S$.

Each set can have its own moat but all moats must be disjoint.

An edge cannot be shorter than all the moats that it has to cross.

---

**Algorithm 1** PrimalDualShortestPath

1: $y \leftarrow 0$
2: $F \leftarrow \emptyset$
3: **while** there is no $s$-$t$ path in $(V, F)$ **do**
4:      Let $C$ be the connected component of $(V, F)$ containing $s$
5:      Increase $y_C$ until there is an edge $e' \in \delta(C)$ such that $\sum_{S:e' \in \delta(S)} y_S = c(e')$.
6:      $F \leftarrow F \cup \{e'\}$
7: Let $P$ be an $s$-$t$ path in $(V, F)$
8: **return** $P$

---

**Lemma 100**

*At each point in time the set $F$ forms a tree.*

**Proof:**

- ▶ In each iteration we take the current connected component from $(V, F)$ that contains $s$ (call this component $C$) and add some edge from $\delta(C)$ to $F$.
- ▶ Since, at most one end-point of the new edge is in $C$ the edge cannot close a cycle.

---

$$\sum_{e \in P} c(e) = \sum_{e \in P} \sum_{S:e \in \delta(S)} y_S$$
$$= \sum_{S:s \in S, t \notin S} |P \cap \delta(S)| \cdot y_S \ .$$

If we can show that $y_S > 0$ implies $|P \cap \delta(S)| = 1$ gives

$$\sum_{e \in P} c(e) = \sum_S y_S \leq \text{OPT}$$

by weak duality.

Hence, we find a shortest path.

If $S$ contains two edges from $P$ then there must exist a subpath $P'$ of $P$ that starts and ends with a vertex from $S$ (and all interior vertices are not in $S$).

When we increased $y_S$, $S$ was a connected component of the set of edges $F'$ that we had chosen till this point.

$F' \cup P'$ contains a cycle. Hence, also the final set of edges contains a cycle.

This is a contradiction.

**Steiner Forest Problem:**
Given a graph $G = (V, E)$, together with source-target pairs $s_i, t_i, i = 1, \ldots, k$, and a cost function $c : E \to \mathbb{R}^+$ on the edges. Find a subset $F \subseteq E$ of the edges such that for every $i \in \{1, \ldots, k\}$ there is a path between $s_i$ and $t_i$ only using edges in $F$.

$$
\begin{array}{lrcl}
\min & & \sum_e c(e) x_e & \\
\text{s.t.} \quad \forall S \subseteq V : S \in S_i \text{ for some } i & \sum_{e \in \delta(S)} x_e & \geq & 1 \\
\forall e \in E & x_e & \in & \{0, 1\}
\end{array}
$$

Here $S_i$ contains all sets $S$ such that $s_i \in S$ and $t_i \notin S$.

$$
\begin{array}{lrcl}
\max & & \sum_{S \,:\, \exists i \text{ s.t. } S \in S_i} y_S & \\
\text{s.t.} \quad \forall e \in E & \sum_{S : e \in \delta(S)} y_S & \leq & c(e) \\
& y_S & \geq & 0
\end{array}
$$

The difference to the dual of the shortest path problem is that we have many more variables (sets for which we can generate a moat of non-zero width).

**Algorithm 1** FirstTry

1: $y \leftarrow 0$
2: $F \leftarrow \emptyset$
3: **while** not all $s_i$-$t_i$ pairs connected in $F$ **do**
4:     Let $C$ be some connected component of $(V, F)$ such that $|C \cap \{s_i, t_i\}| = 1$ for some $i$.
5:     Increase $y_C$ until there is an edge $e' \in \delta(C)$ s.t. $\sum_{S \in S_i : e' \in \delta(S)} y_S = c_{e'}$
6:     $F \leftarrow F \cup \{e'\}$
7: **return** $\bigcup_i P_i$

## Slide 453

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S : e \in \delta(S)} y_S = \sum_{S} |\delta(S) \cap F| \cdot y_S \ .$$

If we show that $y_S > 0$ implies that $|\delta(S) \cap F| \le \alpha$ we are in good shape.

However, this is not true:

- ▶ Take a complete graph on $k + 1$ vertices $v_0, v_1, \ldots, v_k$.
- ▶ The $i$-th pair is $v_0$-$v_i$.
- ▶ The first component $C$ could be $\{v_0\}$.
- ▶ We only set $y_{\{v_0\}} = 1$. All other dual variables stay $0$.
- ▶ The final set $F$ contains all edges $\{v_0, v_i\}$, $i = 1, \ldots, k$.
- ▶ $y_{\{v_0\}} > 0$ but $|\delta(\{v_0\}) \cap F| = k$.

## Slide 454

**Algorithm 1** SecondTry

1: $y \leftarrow 0; F \leftarrow \emptyset; \ell \leftarrow 0$
2: **while** not all $s_i$-$t_i$ pairs connected in $F$ **do**
3: $\quad \ell \leftarrow \ell + 1$
4: $\quad$ Let $C$ be set of all connected components $C$ of $(V, F)$ such that $|C \cap \{s_i, t_i\}| = 1$ for some $i$.
5: $\quad$ Increase $y_C$ for all $C \in C$ uniformly until for some edge $e_\ell \in \delta(C'), C' \in C$ s.t. $\sum_{S : e_\ell \in \delta(S)} y_S = c_{e_\ell}$
6: $\quad F \leftarrow F \cup \{e_\ell\}$
7: $F' \leftarrow F$
8: **for** $k \leftarrow \ell$ downto $1$ **do** // reverse deletion
9: $\quad$ **if** $F' - e_k$ is feasible solution **then**
10: $\quad\quad$ remove $e_k$ from $F'$
11: **return** $F'$

## Slide 455

The reverse deletion step is not strictly necessary this way. It would also be sufficient to simply delete all unnecessary edges in any order.

## Slide 456

# Example
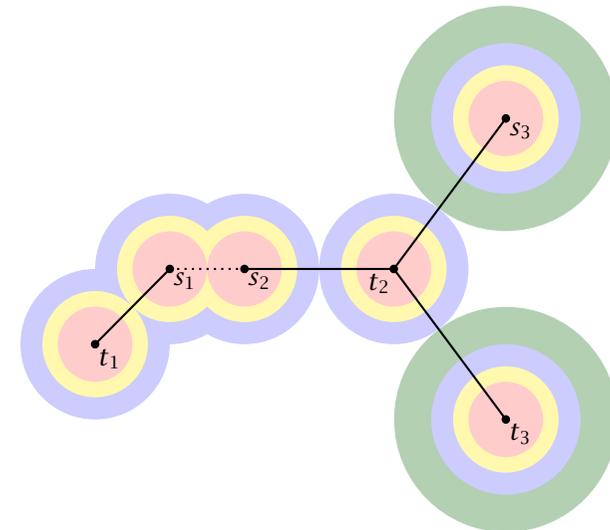
**Lemma 101**

*For any $C$ in any iteration of the algorithm*

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \le 2|C|$$

This means that the number of times a moat from $\mathcal{C}$ is crossed in the final solution is at most twice the number of moats.

**Proof:** later...

---

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} y_S = \sum_{S} |F' \cap \delta(S)| \cdot y_S .$$

We want to show that

$$\sum_{S} |F' \cap \delta(S)| \cdot y_S \le 2 \sum_{S} y_S$$

▶ In the $i$-th iteration the increase of the left-hand side is

$$\epsilon \sum_{C \in \mathcal{C}} |F' \cap \delta(C)|$$

and the increase of the right hand side is $2\epsilon|C|$.

▶ Hence, by the previous lemma the inequality holds after the iteration if it holds in the beginning of the iteration.

---

**Lemma 102**

*For any set of connected components $\mathcal{C}$ in any iteration of the algorithm*

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \le 2|C|$$

**Proof:**

▶ At any point during the algorithm the set of edges forms a forest (why?).

▶ Fix iteration $i$. $e_i$ is the set we add to $F$. Let $F_i$ be the set of edges in $F$ at the beginning of the iteration.

▶ Let $H = F' - F_i$.

▶ All edges in $H$ are necessary for the solution.

---

▶ Contract all edges in $F_i$ into single vertices $V'$.

▶ We can consider the forest $H$ on the set of vertices $V'$.

▶ Let $\deg(v)$ be the degree of a vertex $v \in V'$ within this forest.

▶ Color a vertex $v \in V'$ red if it corresponds to a component from $\mathcal{C}$ (an active component). Otw. color it blue. (Let $B$ the set of blue vertices (with non-zero degree) and $R$ the set of red vertices)

▶ We have

$$\sum_{v \in R} \deg(v) \ge \sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \overset{?}{\le} 2|C| = 2|R|$$

- Suppose that no node in $B$ has degree one.
- Then

$$\sum_{v \in R} \deg(v) = \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v)$$
$$\leq 2(|R| + |B|) - 2|B| = 2|R|$$

- Every blue vertex with non-zero degree must have degree at least two.
  - Suppose not. The single edge connecting $b \in B$ comes from $H$, and, hence, is necessary.
  - But this means that the cluster corresponding to $b$ must separate a source-target pair.
  - But then it must be a red node.

# 21 Cuts & Metrics

**Shortest Path**

$$
\begin{array}{lllll}
\min & & \sum_e c(e) x_e & & \\
\text{s.t.} & \forall S \in S & \sum_{e \in \delta(S)} x_e & \geq & 1 \\
& \forall e \in E & x_e & \in & \{0,1\}
\end{array}
$$

$S$ is the set of subsets that separate $s$ from $t$.

**The Dual:**

$$
\begin{array}{lllll}
\max & & \sum_S y_S & & \\
\text{s.t.} & \forall e \in E & \sum_{S:e \in \delta(S)} y_S & \leq & c(e) \\
& \forall S \in S & y_S & \geq & 0
\end{array}
$$

The Separation Problem for the Shortest Path LP is the Minimum Cut Problem.

# 21 Cuts & Metrics

**Minimum Cut**

$$
\begin{array}{lllll}
\min & & \sum_e c(e) x_e & & \\
\text{s.t.} & \forall P \in \mathcal{P} & \sum_{e \in P} x_e & \geq & 1 \\
& \forall e \in E & x_e & \in & \{0,1\}
\end{array}
$$

$\mathcal{P}$ is the set of path that connect $s$ and $t$.

**The Dual:**

$$
\begin{array}{lllll}
\max & & \sum_P y_P & & \\
\text{s.t.} & \forall e \in E & \sum_{P:e \in P} y_P & \leq & c(e) \\
& \forall P \in \mathcal{P} & y_P & \geq & 0
\end{array}
$$

The Separation Problem for the Minimum Cut LP is the Shortest Path Problem.

# 21 Cuts & Metrics

**Observations:**

Suppose that $\ell_e$-values are solution to Minimum Cut LP.
- We can view $\ell_e$ as defining the length of an edge.
- Define $d(u, v) = \min_{\text{path } P \text{ btw. } u \text{ and } v} \sum_{e \in P} \ell_e$ as the Shortest Path Metric induced by $\ell_e$.
- We have $d(u, v) = \ell_e$ for every edge $e = (u, v)$, as otw. we could reduce $\ell_e$ without affecting the distance between $s$ and $t$.

**Remark for bean-counters:**
$d$ is not a metric on $V$ but a semimetric as two nodes $u$ and $v$ could have distance zero.

**How do we round the LP?**

- Let $B(s, r)$ be the ball of radius $r$ around $s$ (w.r.t. metric $d$). Formally:
$$B = \{v \in V \mid d(s, v) \le r\}$$

- For $0 \le r < 1$, $B(s, r)$ is an $s$-$t$-cut.

**Which value of $r$ should we choose? choose randomly!!!**

Formally:
choose $r$ u.a.r. (uniformly at random) from interval $[0, 1)$

---

**What is the probability that an edge $(u, v)$ is in the cut?**
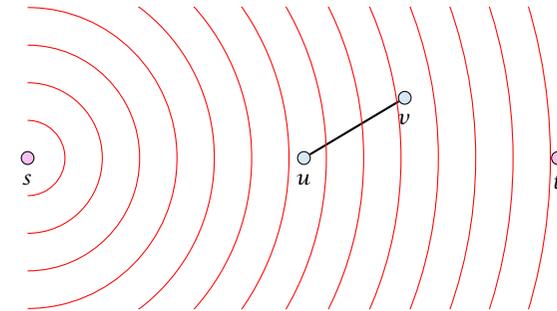


- asssume wlog. $d(s, u) \le d(s, v)$

$$\Pr[e \text{ is cut}] = \Pr[r \in [d(s, u), d(s, v))] \le \frac{d(s, v) - d(s, u)}{1 - 0}$$
$$\le \ell_e$$

---

**What is the expected size of a cut?**

$$\mathrm{E}[\text{size of cut}] = \mathrm{E}\Big[\sum_e c(e) \Pr[e \text{ is cut}]\Big]$$
$$\le \sum_e c(e) \ell_e$$

On the other hand:

$$\sum_e c(e) \ell_e \le \text{size of mincut}$$

as the $\ell_e$ are the solution to the Mincut LP *relaxation*.

Hence, our rounding gives an optimal solution.

---

**Minimum Multicut:**
Given a graph $G = (V, E)$, together with source-target pairs $s_i, t_i$, $i = 1, \ldots, k$, and a capacity function $c : E \to \mathbb{R}^+$ on the edges.
Find a subset $F \subseteq E$ of the edges such that all $s_i$-$t_i$ pairs lie in different components in $G = (V, E \setminus F)$.

$$
\begin{array}{lll}
\min & \sum_e c(e) \ell_e & \\
\text{s.t.} \quad \forall P \in \mathcal{P}_i \text{ for some } i & \sum_{e \in P} \ell_e & \ge \quad 1 \\
\forall e \in E & \ell_e & \in \quad \{0, 1\}
\end{array}
$$

Here $\mathcal{P}_i$ contains all path $P$ between $s_i$ and $t_i$.

**Re-using the analysis for the single-commodity case is difficult.**

$$\Pr[e \text{ is cut}] \leq \text{?}$$

▶ If for some $R$ the balls $B(s_i, R)$ are disjoint between different sources, we get a $1/R$ approximation.

▶ However, this cannot be guaranteed.

---

▶ Assume for simplicity that all edge-length $\ell_e$ are multiples of $\delta \ll 1$.

▶ Replace the graph $G$ by a graph $G'$, where an edge of length $\ell_e$ is replaced by $\ell_e/\delta$ edges of length $\delta$.

▶ Let $B(s_i, z)$ be the ball in $G'$ that contains nodes $v$ with distance $d(s_i, v) \leq z\delta$.

**Algorithm 1** RegionGrowing($s_i, p$)
1: $z \leftarrow 0$
2: **repeat**
3:     flip a coin ($\Pr[\text{heads}] = p$)
4:     $z \leftarrow z + 1$
5: **until** heads
6: **return** $B(s_i, z)$

---

**Algorithm 1** Multicut($G'$)
1: **while** $\exists s_i$-$t_i$ pair in $G'$ **do**
2:     $C \leftarrow$ RegionGrowing($s_i, p$)
3:     $G' = G' \setminus C$ // cuts edges leaving $C$
4: **return** $B(s_i, z)$

▶ probability of cutting an edge is only $p$

▶ a source either does not reach an edge during Region Growing; then it is not cut

▶ if it reaches the edge then it either cuts the edge or protects the edge from being cut by other sources

▶ if we choose $p = \delta$ the probability of cutting an edge is only its LP-value; our expected cost are at most OPT.

---

**Problem:**
We may not cut all source-target pairs.

A component that we remove may contain an $s_i$-$t_i$ pair.

If we ensure that we cut before reaching radius $1/2$ we are in good shape.

- choose $p = 6 \ln k \cdot \delta$
- we make $\frac{1}{2\delta}$ trials before reaching radius $1/2$.
- we say a Region Growing is not successful if it does not terminate before reaching radius $1/2$.

$$\Pr[\text{not successful}] \leq (1-p)^{\frac{1}{2\delta}} = \left((1-p)^{1/p}\right)^{\frac{p}{2\delta}} \leq e^{-\frac{p}{2\delta}} \leq \frac{1}{k^3}$$

- Hence,
$$\Pr[\exists i \text{ that is not successful}] \leq \frac{1}{k^2}$$

---

**What is expected cost?**

$$\mathrm{E}[\text{cutsize}] = \Pr[\text{success}] \cdot \mathrm{E}[\text{cutsize} \mid \text{success}]$$
$$+ \Pr[\text{no success}] \cdot \mathrm{E}[\text{cutsize} \mid \text{no success}]$$

$$\mathrm{E}[\text{cutsize} \mid \text{succ.}] = \frac{\mathrm{E}[\text{cutsize}] - \Pr[\text{no succ.}] \cdot \mathrm{E}[\text{cutsize} \mid \text{no succ.}]}{\Pr[\text{success}]}$$

$$\leq \frac{\mathrm{E}[\text{cutsize}]}{\Pr[\text{success}]} \leq \frac{1}{1 - \frac{1}{k^2}} 6 \ln k \cdot \mathrm{OPT} \leq 8 \ln k \cdot \mathrm{OPT}$$

Note: success means all source-target pairs separated

We assume $k \geq 2$.

---

If we are not successful we simply perform a trivial $k$-approximation.

This only increases the expected cost by at most $\frac{1}{k^2} \cdot k\mathrm{OPT} \leq \mathrm{OPT}/k$.

Hence, our final cost is $\mathcal{O}(\ln k) \cdot \mathrm{OPT}$ in expectation.

---

# Facility Location

Given a set $L$ of (possible) locations for placing facilities and a set $D$ of customers together with cost functions $s : D \times L \to \mathbb{R}^+$ and $o : L \to \mathbb{R}^+$ find a set of facility locations $F$ together with an assignment $\phi : D \to F$ of customers to open facilities such that

$$\sum_{f \in F} o(f) + \sum_c s(c, \phi(c))$$

is minimized.

In the metric facility location problem we have

$$s(c, f) \leq s(c, f') + s(c', f) + s(c', f') \ .$$

## Facility Location

**Integer Program**

$$
\begin{array}{llrcl}
\min & & \sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in D} c_{ij} x_{ij} & & \\
\text{s.t.} & \forall j \in D & \sum_{i\in F} x_{ij} & = & 1 \\
& \forall i \in F, j \in D & x_{ij} & \le & y_i \\
& \forall i \in F, j \in D & x_{ij} & \in & \{0,1\} \\
& \forall i \in F & y_i & \in & \{0,1\}
\end{array}
$$

As usual we get an LP by relaxing the integrality constraints.

## Facility Location

**Dual Linear Program**

$$
\begin{array}{llrcl}
\max & & \sum_{j\in D} v_j & & \\
\text{s.t.} & \forall i \in F & \sum_{j\in D} w_{ij} & \le & f_i \\
& \forall i \in F, j \in D & v_j - w_{ij} & \le & c_{ij} \\
& \forall i \in F, j \in D & w_{ij} & \ge & 0
\end{array}
$$

## Facility Location

**Definition 103**
Given an LP solution $(x^*, y^*)$ we say that facility $i$ neighbours client $j$ if $x_{ij} > 0$. Let $N(j) = \{i \in F : x_{ij}^* > 0\}$.

**Lemma 104**
*If $(x^*, y^*)$ is an optimal solution to the facility location LP and $(v^*, w^*)$ is an optimal dual solution, then $x_{ij}^* > 0$ implies $c_{ij} \le v_j^*$.*

Follows from slackness conditions.

Suppose we open set $S \subseteq F$ of facilities s.t. for all clients we have $S \cap N(j) \neq \emptyset$.

Then every client $j$ has a facility $i$ s.t. assignment cost for this client is at most $c_{ij} \leq v_j^*$.

Hence, the total assignment cost is

$$\sum_j c_{i_j j} \leq \sum_j v_j^* \leq \text{OPT} ,$$

where $i_j$ is the facility that client $j$ is assigned to.

**Problem: Facility cost may be huge!**

Suppose we can partition a subset $F' \subseteq F$ of facilities into neighbour sets of some clients. I.e.

$$F' = \biguplus_k N(j_k)$$

where $j_1, j_2, \ldots$ form a subset of the clients.

Now in each set $N(j_k)$ we open the cheapest facility. Call it $f_{i_k}$.

We have

$$f_{i_k} = f_{i_k} \sum_{i \in N(j_k)} x_{ij_k}^* \leq \sum_{i \in N(j_k)} f_i x_{ij_k}^* \leq \sum_{i \in N(j_k)} f_i y_i^* .$$

Summing over all $k$ gives

$$\sum_k f_{i_k} \leq \sum_k \sum_{i \in N(j_k)} f_i y_i^* = \sum_{i \in F'} f_i y_i^* \leq \sum_{i \in F} f_i y_i^*$$

Facility cost is at most the facility cost in an optimum solution.

**Problem: so far clients $j_1, j_2, \ldots$ have a neighboring facility. What about the others?**

### Definition 105
Let $N^2(j)$ denote all neighboring clients of the neighboring facilities of client $j$.

Note that $N(j)$ is a set of facilities while $N^2(j)$ is a set of clients.

## Slide 485

**Algorithm 1** FacilityLocation

1: $C \leftarrow D$ // unassigned clients
2: $k \leftarrow 0$
3: **while** $C \neq 0$ **do**
4:     $k \leftarrow k + 1$
5:     choose $j_k \in C$ that minimizes $v_j^*$
6:     choose $i_k \in N(j_k)$ as cheapest facility
7:     assign $j_k$ and all unassigned clients in $N^2(j_k)$ to $i_k$
8:     $C \leftarrow C - \{j_k\} - N^2(j_k)$

## Slide 486

Facility cost of this algorithm is at most OPT because the sets $N(j_k)$ are disjoint.

**Total assignment cost:**

▶ Fix $k$; set $j = j_k$ and $i = i_k$. We know that $c_{ij} \leq v_j^*$.

▶ Let $\ell \in N^2(j)$ and $h$ (one of) its neighbour(s) in $N(j)$.

$$c_{i\ell} \leq c_{ij} + c_{hj} + c_{h\ell} \leq v_j^* + v_j^* + v_\ell^* \leq 3v_\ell^*$$

Summing this over all facilities gives that the total assignment cost is at most $3 \cdot \text{OPT}$. Hence, we get a $4$-approximation.

## Slide 487

In the above analysis we use the inequality

$$\sum_{i \in F} f_i y_i^* \leq \text{OPT} .$$

We know something stronger namely

$$\sum_{i \in F} f_i y_i^* + \sum_{i \in F} \sum_{j \in D} c_{ij} x_{ij}^* \leq \text{OPT} .$$

## Slide 488

**Observation:**

▶ Suppose when choosing a client $j_k$, instead of opening the cheapest facility in its neighborhood we choose a random facility according to $x_{ij_k}^*$.

▶ Then we incur connection cost

$$\sum_i c_{ij_k} x_{ij_k}^*$$

for client $j_k$. (In the previous algorithm we estimated this by $v_{j_k}^*$).

▶ Define

$$C_j^* = \sum_i c_{ij} x_{ij}^*$$

to be the connection cost for client $j$.

**What will our facility cost be?**

We only try to open a facility once (when it is in neighborhood of some $j_k$). (recall that neighborhoods of different $j_k's$ are disjoint).

We open facility $i$ with probability $x_{ij_k} \leq y_i$ (in case it is in some neighborhood; otw. we open it with probability zero).

Hence, the expected facility cost is at most

$$\sum_{i \in F} f_i y_i .$$

---

**Algorithm 1** FacilityLocation
1: $C \leftarrow D$ // unassigned clients
2: $k \leftarrow 0$
3: **while** $C \neq 0$ **do**
4:      $k \leftarrow k + 1$
5:      choose $j_k \in C$ that minimizes $v_j^* + C_j^*$
6:      choose $i_k \in N(j_k)$ according to probability $x_{ij_k}$.
7:      assign $j_k$ and all unassigned clients in $N^2(j_k)$ to $i_k$
8:      $C \leftarrow C - \{j_k\} - N^2(j_k)$

---

**Total assignment cost:**

- Fix $k$; set $j = j_k$.
- Let $\ell \in N^2(j)$ and $h$ (one of) its neighbour(s) in $N(j)$.
- If we assign a client $\ell$ to the same facility as $i$ we pay at most

$$\sum_i c_{ij} x_{ij_k}^* + c_{hj} + c_{h\ell} \leq C_j^* + v_j^* + v_\ell^* \leq C_\ell^* + 2v_\ell^*$$

Summing this over all clients gives that the total assignment cost is at most

$$\sum_j C_j^* + \sum_j 2v_j^* \leq \sum_j C_j^* + 2\text{OPT}$$

Hence, it is at most $2\text{OPT}$ plus the total assignment cost in an optimum solution.

Adding the facility cost gives a $3$-approximation.

---