Linear Programming and simplex algorithm

1 What is Linear Programming?

Generally speaking, all problems with linear objective and linear equalities/inequalities constraints could be considered as Linear Programming. However, there are some widely accepted formulations.

$$\boxed{\begin{aligned} &\min c^T x \\ &Ax=b \\ &x \geq 0 \end{aligned}}$$

Standard FORM

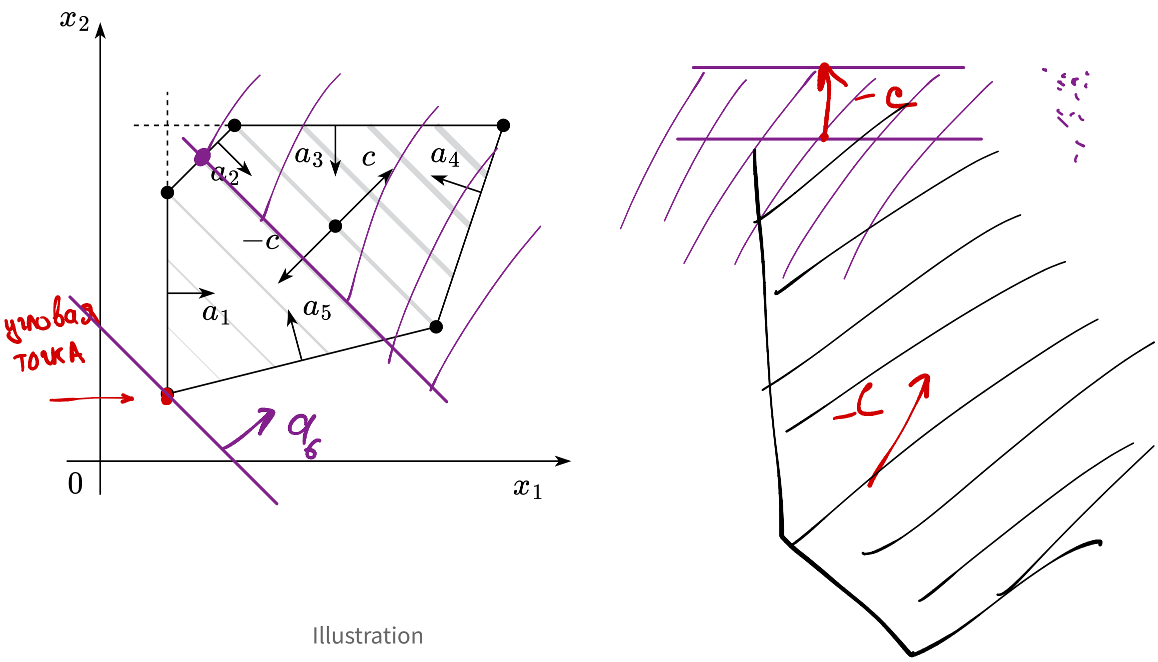
$m \times n$

$$\boxed{\begin{aligned} &\min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } & Ax \leq b \end{aligned}} \qquad A \in \mathbb{R}^{m \times n} \tag{LP.Basic}$$

$m > n \quad S = \varnothing$

$m = n \quad x^* = A^{-1} b$

$m < n$



Illustration

for some vectors $c \in \mathbb{R}^n, b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times n}$. Where the inequalities are interpreted component-wise.

Standard form. This form seems to be the most intuitive and geometric in terms of visualization. Let us have vectors $c \in \mathbb{R}^n, b \in \mathbb{R}^m$ and matrix $A \in \mathbb{R}^{m \times n}$.

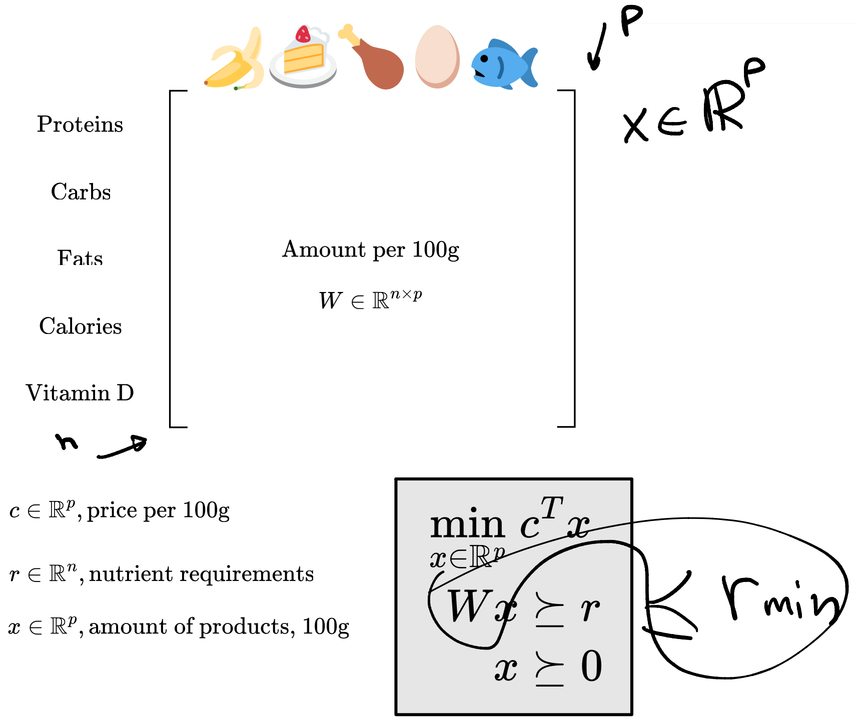
$$\begin{aligned} &\min_{x \in \mathbb{R}^n} c^\top x \\ \text{s.t. } & Ax = b \\ & x_i \geq 0, \; i = 1, \dots, n \end{aligned} \tag{LP.Standard}$$

2 Examples of LP problems

Diet problem

Imagine, that you have to construct a diet plan from some set of products: 🍌🍰🍗🥚🐟. Each of the products has its own vector of nutrients. Thus, all the food information could be processed through the matrix W. Let also assume, that we have the vector of requirements for each of nutrients $r \in \mathbb{R}^n$. We need to find the cheapest configuration of the diet, which meets all the requirements:

$$\begin{aligned} &\min_{x \in \mathbb{R}^p} c^\top x \\ \text{s.t. } & Wx \geq r \\ & x_i \geq 0, \; i = 1, \dots, n \end{aligned}$$



Proteins

Carbs

Fats

Calories

Vitamin D

Amount per 100g

$W \in \mathbb{R}^{n \times p}$

$c \in \mathbb{R}^p$, price per 100g

$r \in \mathbb{R}^n$, nutrient requirements

$x \in \mathbb{R}^p$, amount of products, 100g

$$\boxed{\begin{aligned} \min_{x \in \mathbb{R}^p} \ & c^T x \\ W x &\succeq r \\ x &\succeq 0 \end{aligned}}$$

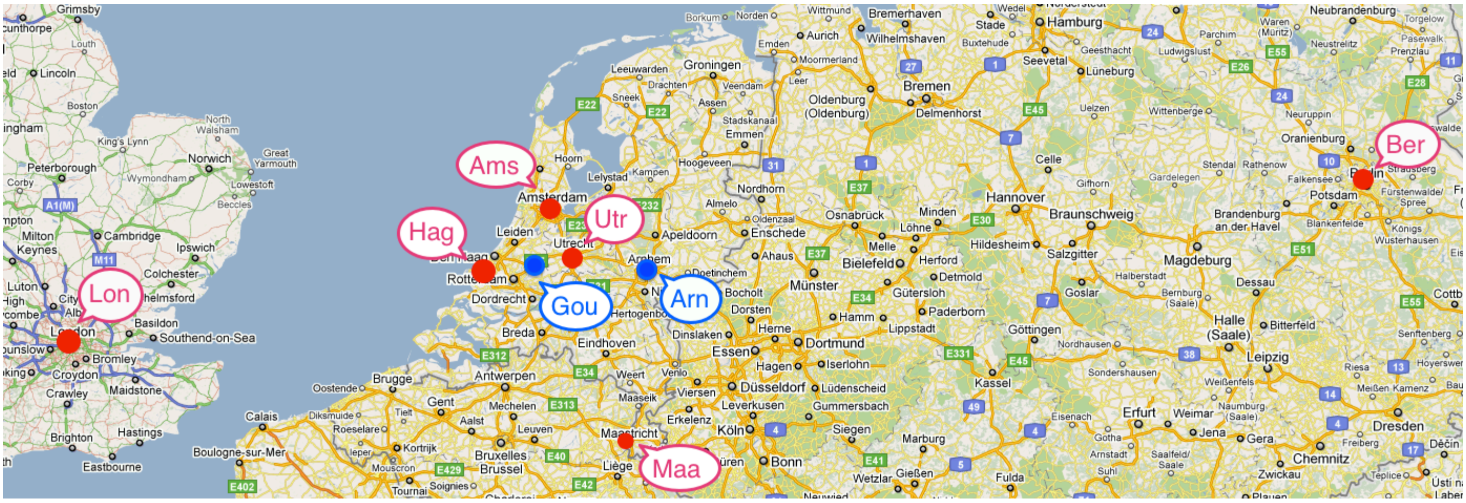
Illustration

Transportation problem

Source

The prototypical transportation problem deals with the distribution of a commodity from a set of sources to a set of destinations. The object is to minimize total transportation costs while satisfying constraints on the supplies available at each of the sources, and satisfying demand requirements at each of the destinations.

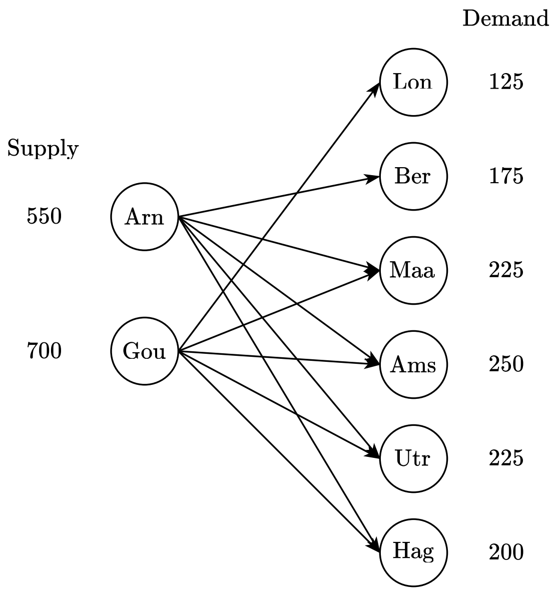
Here we illustrate the transportation problem using an example from Chapter 5 of Johannes Bisschop, “AIMMS Optimization Modeling”, Paragon Decision Sciences, 1999. In this example there are two factories and six customer sites located in 8 European cities as shown in the following map. The customer sites are labeled in red, the factories are labeled in blue.



West Europe Map

Customer	Arnhem [€/ton]	Gouda [€/ton]	Demand [tons]
London	n/a	2.5	125
Berlin	2.5	n/a	175
Maastricht	1.6	2.0	225
Amsterdam	1.4	1.0	250
Utrecht	0.8	1.0	225
The Hague	1.4	0.8	200
Supply [tons]	550 tons	700 tons	

This can be represented as the following graph:



Graph associated with the problem

For each link we can have a parameter $T[c, s]$ denoting the cost of shipping a ton of goods over the link. What we need to determine is the amount of goods to be shipped over each link, which we will represent as a non-negative decision variable $x[c, s]$.
The problem objective is to minimize the total shipping cost to all customers from all sources.

$$\text{minimize:} \quad \boxed{\text{Cost} = \sum_{c \in Customers} \sum_{s \in Sources} T[c, s] x[c, s]}$$

Shipments from all sources can not exceed the manufacturing capacity of the source.

$$\boxed{\sum_{c \in Customers} x[c, s] \leq \text{Supply}[s] \qquad \forall s \in Sources}$$

Shipments to each customer must satisfy their demand.

$$\boxed{\sum_{s \in Sources} x[c, s] = \text{Demand}[c] \qquad \forall c \in Customers}$$

The code for the problem is available here: 💻

Blending problem

Source
A brewery receives an order for 100 gallons of 4% ABV (alchohol by volume) beer. The brewery has on hand beer Okhota that is 4.5% ABV that cost USD 0.32 per gallon to make, and beer Baltos that is 3.7% ABV and cost USD 0.25 per gallon. Water could also be used as a blending agent at a cost of USD 0.05 per gallon. Find the minimum cost blend that meets the customer requirements.

2.0.1 Model Formulation

2.0.1.1 Objective Function

If we let subscript c denote a blending component from the set of blending components C, and denote the volume of c used in the blend as x_c, the cost of the blend is

$$\boxed{\text{cost} = \sum_{c \in C} \overset{\text{кол-во}}{x_c} \overset{\text{цена}}{P_c}}$$

where P_c is the price per unit volume of c. Using the Python data dictionary defined above, the price P_c is given by `data[c]['cost']`.

2.0.1.2 Volume Constraint

The customer requirement is produce a total volume V. Assuming ideal solutions, the constraint is given by

$$\boxed{V = \sum_{c \in C} x_c}$$

where x_c denotes the volume of component c used in the blend.

2.0.1.3 Product Composition Constraint

The product composition is specified as 4% alchohol by volume. Denoting this as $\bar{A}$, the constraint may be written as

$$\boxed{\bar{A} = \frac{\sum_{c\in C} x_c A_c}{\sum_{c\in C} x_c}}$$

where A_c is the alcohol by volume for component c. As written, this is a nonlinear constraint. Multiplying both sides of the equation by the denominator yields a linear constraint

$$\boxed{\bar{A}\sum_{c\in C} x_c = \sum_{c\in C} x_c A_c}$$

A final form for this constraint can be given in either of two versions. In the first version we subtract the left-hand side from the right to give

$$0 = \sum_{c\in C} x_c \left(A_c - \bar{A}\right) \;\; \text{Version 1 of the linear blending constraint}$$

Alternatively, the summation on the left-hand side corresponds to total volume. Since that is known as part of the problem specification, the blending constraint could also be written as

$$\bar{A}V = \sum_{c\in C} x_c A_c \;\text{Version 2 of the linear blending constraint}$$

Which should you use? Either will generally work well. The advantage of version 1 is that it is fully specified by a product requirement $\bar{A}$, which is sometimes helpful in writing elegant Python code.

The code for the problem is available here: 💻

3 Basic transformations

Inequality to equality by increasing the dimension of the problem by m.

$$\boxed{Ax \leq b \leftrightarrow \begin{cases} Ax + z = b \\ z \geq 0 \end{cases}}$$

unsigned variables to nonnegative variables.

$$\boxed{x \leftrightarrow \begin{cases} x = x_+ - x_- \\ x_+ \geq 0 \\ x_- \geq 0 \end{cases}}$$

Chebyshev approximation problem

$$\boxed{\min_{x\in\mathbb{R}^n} \|Ax - b\|_\infty \leftrightarrow \min_{x\in\mathbb{R}^n} \max_i |a_i^\top x - b_i|}$$

Could be equivalently written as a LP with thre replacement of maximum coordinate of a vector: $c\begin{pmatrix}0\\ \vdots \\ 1\end{pmatrix}\begin{pmatrix}x\\ t\end{pmatrix}$

$$\boxed{\begin{aligned} &\min_{t\in\mathbb{R}, x\in\mathbb{R}^n} t \\ \text{s.t. } & a_i^\top x - b_i \leq t, \; i = 1, \ldots, n \\ & - a_i^\top x + b_i \leq t, \; i = 1, \ldots, n \end{aligned}}$$

i-ая коорд.

$Ax-b$

$-t \leq \quad \leq t$

l_1 approximation problem

$$\boxed{\min_{x\in\mathbb{R}^n} \|Ax - b\|_1 \leftrightarrow \min_{x\in\mathbb{R}^n} \sum_{i=1}^n |a_i^\top x - b_i|}$$

Could be equivalently written as a LP with thre replacement of the sum of coordinates of a vector:

? ? ?

$|a_i^T x - b_i| \le t_i$

$$
\begin{aligned}
\min_{t\in\mathbb{R}^n, x\in\mathbb{R}^n} \; & \mathbf{1}^\top t = \sum t_i \\
\text{s.t. } & a_i^\top x - b_i \le t_i, \; i = 1,\dots,n \\
& -a_i^\top x + b_i \le t_i, \; i = 1,\dots,n
\end{aligned}
$$

$-t \le a_i^T x - b_i \le t$

$-a_i^T x + b_i \le t_i$

4 Duality

There are four possibilities:

- Both the primal and the dual are infeasible.
- The primal is infeasible and the dual is unbounded.
- The primal is unbounded and the dual is infeasible.
- Both the primal and the dual are feasible and their optimal values are equal.

Example

Ensure, that the following standard form LP:

$c^T x = b^T y$

$Ax = b$

p^*

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \; & c^\top x \\
\text{s.t. } & Ax = b \\
& x_i \ge 0, \; i = 1,\dots,n
\end{aligned}
$$

Has the following dual:

d^*

$$
\begin{aligned}
\max_{y\in\mathbb{R}^n} \; & b^\top y \\
\text{s.t. } & A^T y \preceq c
\end{aligned}
$$

Find the dual problem to the problem above (it should be the original LP).

$L = c^T x + \nu^T(Ax - b) - \lambda^T x$

$g = \inf_x L = -b^T \nu$

$c + A^T \nu - \lambda = 0$

$\lambda \ge 0$

Dual PROBLEM:

$-b^T \nu \to \max_{\nu, \lambda}$

$c + A^T \nu = \lambda$

$\lambda \ge 0$

5 Idea of simplex algorithm

- The Simplex Algorithm walks along the edges of the polytope, at every corner choosing the edge that decreases $c^\top x$ most
- This either terminates at a corner, or leads to an unconstrained edge ($-\infty$ optimum)

$-b^T \nu \to \max_{\nu}$

$c + A^T \nu \ge 0$

$-\nu = y$

We will illustrate simplex algorithm for the simple inequality form of LP:

$$
\begin{aligned}
\min_{x\in\mathbb{R}^n} \; & c^\top x \\
\text{s.t. } & Ax \le b
\end{aligned}
\qquad \text{(LP.Inequality)}
$$

Опр. 1

Definition: a **basis** B is a subset of n (integer) numbers between 1 and m, so that $\operatorname{rank} A_B = n$. Note, that we can associate submatrix A_B and corresponding right-hand side b_B with the basis B. Also, we can derive a point of intersection of all these hyperplanes from basis: $x_B = A_B^{-1} b_B$

Опр. 2

If $Ax_B \le b$, then basis B is **feasible**.

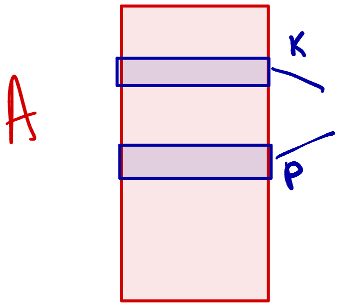
допустимый

$X := X_B = A_B^{-1} b_B$

A basis B is optimal if x_B is an optimum of the LP.Inequality.

Опр. 3

$\underset{2\times n}{A_B} \cdot \underset{n\times 1}{X} = \underset{2\times 1}{b_B}$

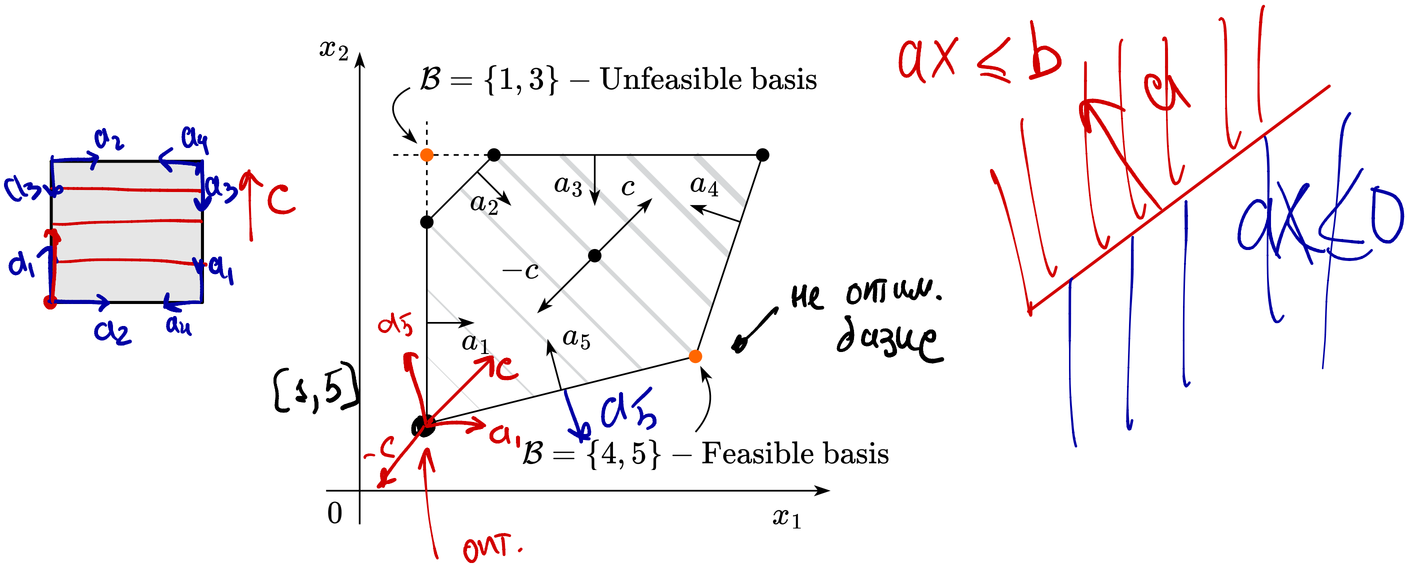


$B = \{k, p\} = \{a_k, a_p\}$

$c^T x \to \min$

$Ax \le b$

$B = \{.....\}$ - базис $\Rightarrow$ $A_B \cdot X = b_B$ $\quad ax \geq 0$



Illustration

Theorem

1. If Standartd LP has a nonempty feasible region, then there is at least one basic feasible point
2. If Standartd LP has solutions, then at least one such solution is a basic optimal point.
3. If Standartd LP is feasible and bounded, then it has an optimal solution.

Since we have a basis, we can decompose our objective vector c in this basis and find the scalar coefficients λ_B: разл. c в базисе $A_{B_{n \times n}}$

X_B задаётся $A_B X_B = b_B$

$$\boxed{\lambda_B^\top A_B = -c^\top} \leftrightarrow \boxed{\lambda_B^\top = -c^\top A_B^{-1}}$$

Theorem

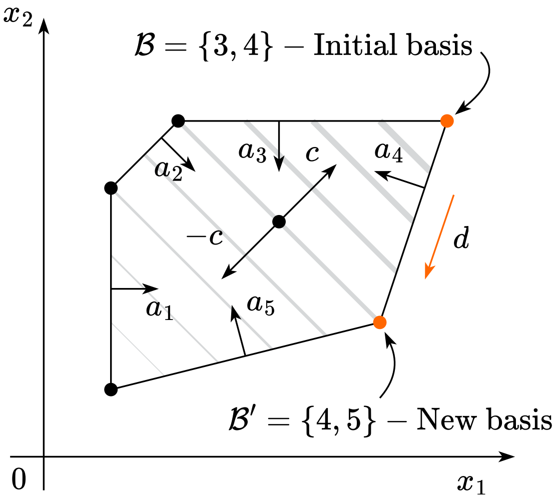
If all components of λ_B are non-positive and B is feasible, then B is optimal.

🔽 **Proof**

5.1 Changing basis

Suppose, some of the coefficients of λ_B are positive. Then we need to go through the edge of the polytope to the new vertex (i.e., switch the basis)

Вопрос
выбора
стартовой
угловой
точки
нетривиаль



x_2
$\mathcal{B} = \{3, 4\}$ – Initial basis
a_3
c
a_4
a_2
$-c$
d
a_1
a_5
$\mathcal{B}' = \{4, 5\}$ – New basis
0
x_1

Illustration

$$
x_{B'} = x_B + \mu d = A^{-1}_{B'} b_{B'}
$$

5.2 Finding an initial basic feasible solution

Let us consider LP.Canonical.

$$
\begin{split}
&\min_{x \in \mathbb{R}^n} c^\top x \\
\text{s.t. } & Ax = b \\
& x_i \geq 0, \; i = 1, \dots, n
\end{split}
$$

The proposed algorithm requires an initial basic feasible solution and corresponding basis. To compute this solution and basis, we start by multiplying by -1 any row i of $Ax = b$ such that $b_i < 0$. This ensures that $b \geq 0$. We then introduce artificial variables $z \in \mathbb{R}^m$ and consider the following LP:

$$
\begin{split}
&\min_{x \in \mathbb{R}^n, z \in \mathbb{R}^m} 1^\top z \\
\text{s.t. } & Ax + Iz = b \\
& x_i, z_j \geq 0, \; i = 1, \dots, n \; j = 1, \dots, m
\end{split}
\tag{LP.Phase 1}
$$

which can be written in canonical form $\min\{\tilde{c}^\top \tilde{x} \mid \tilde{A}\tilde{x} = \tilde{b}, \tilde{x} \geq 0\}$ by setting

$$
\tilde{x} = \begin{bmatrix} x \\ z \end{bmatrix}, \quad \tilde{A} = [A \; I], \quad \tilde{b} = b, \quad \tilde{c} = \begin{bmatrix} 0_n \\ 1_m \end{bmatrix}
$$

An initial basis for LP.Phase 1 is $\tilde{A}_B = I, \tilde{A}_N = A$ with corresponding basic feasible solution $\tilde{x}_N = 0, \tilde{x}_B = \tilde{A}^{-1}_B \tilde{b} = \tilde{b} \geq 0$. We can therefore run the simplex method on LP.Phase 1, which will converge to an optimum $\tilde{x}^*$. $\tilde{x} = (\tilde{x}_N \; \tilde{x}_B)$. There are several possible outcomes:

- $\tilde{c}^\top \tilde{x} > 0$. Original primal is infeasible.
- $\tilde{c}^\top \tilde{x} = 0 \to 1^\top z^* = 0$. The obtained solution is a start point for the original problem (probably with slight modification).

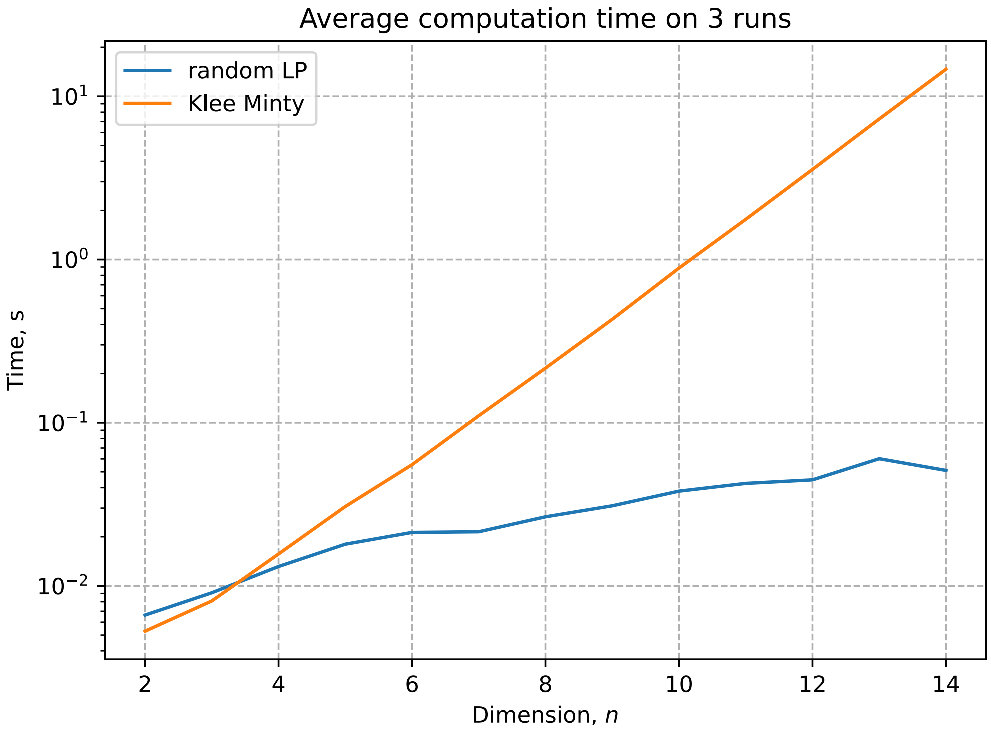
6 Convergence

Since the number of edge point is finite, algorithm should converge (except some degenerate cases, which are not covered here). However, the convergence could be exponentially slow, due to the high number of edges. There is the following iconic example, when simplex algorithm should perform exactly all vertexes.

6.1 Klee Minty example

In the following problem simplex algorithm needs to check $2^n - 1$ vertexes with $x_0 = 0$.

$$
\begin{aligned}
& \max_{x \in \mathbb{R}^n} 2^{n-1}x_1 + 2^{n-2}x_2 + \dots + 2x_{n-1} + x_n \\
\text{s.t. } & x_1 \leq 5 \\
& 4x_1 + x_2 \leq 25 \\
& 8x_1 + 4x_2 + x_3 \leq 125 \\
& \ldots \\
& 2^n x_1 + 2^{n-1}x_2 + 2^{n-2}x_3 + \ldots + x_n \leq 5^n \qquad x \geq 0
\end{aligned}
$$



Illustration

7 Summary

- A wide variety of applications could be formulated as the linear programming.
- Simplex algorithm is simple, but could work exponentially long.
- Khachiyan's ellipsoid method is the first to be proved running at polynomial complexity for LPs. However, it is usually slower than simplex in real problems.
- Interior point methods are the last word in this area. However, good implementations of simplex-based methods and interior point methods are similar for routine applications of linear programming.

8 Code

Open In Colab

9 Materials

- Linear Programming. in V. Lempitsky optimization course.
- Simplex method. in V. Lempitsky optimization course.
- Overview of different LP solvers
- TED talks watching optimization

- Overview of ellipsoid method
- Comprehensive overview of linear programming
- Converting LP to a standard form