Linear Programming and the Birkhoff Polytope

Luis Carlos Mantilla
José Luis Mora

December 7, 2020

1 Introduction

Linear programming (LP) is a mathematical programming field that deals with optimal solutions to a linear problem. It means that linear terms define the problem's restrictions and the cost function we wish to maximize or minimize. We can give a geometrical interpretation of these types of problems. A set of linear constraints define a convex polyhedron, which can be empty, unbounded, or bounded. In the latter scenario, the linear restrictions are satisfied inside a convex polytope.

Additionally, the optimal solution for bounded LP problems is achieved on a vertex of the corresponding polytope. There are several algorithms dedicated to finding the solution to such problems. For example, the simplex algorithm was one of the first proposed algorithms, where the core idea lies in moving only along the edges of the polytope until an optimal vertex is found. Other important algorithms are interior-point methods, which, contrary to the simplex method, search for the optimal vertex optimizing either outside or inside the polytope instead of moving in its boundary. The applicability of these algorithms is immense due to the simplicity of the problems solved.

An example is a family of LP problems called the assignment problems, whose goal is to find the best way to assign n elements of a particular set A to n elements of another set B. The formulation of the assignment problem has a more natural language in network-flow theory. A directed graph with costs and capacities in its edges will represent the assignment problem. Nevertheless, we can solve these problems by employing LP methods. The solution set for such LP problems can be interpreted as a specific polytope called the Birkhoff polytope. This object consists of all the matrices whose rows and columns individually add to one, otherwise known as doubly stochastic matrices. The Birkhoff-von Neuman theorem states that this polytope is the convex hull of the $n \times n$ permutation matrices. Therefore, an optimal solution for this type of LP problems will be a permutation matrix. This text will expose some of the theory behind assignment problems and apply the machinery already developed for finding the best energy-efficient way of dispatching elevators on a busy building.

2 Preliminaries

We expose some of the basic definitions used throughout the text. We do not intend to give an extensive review of the preliminaries. For further details, we direct the reader to Ziegler's book [1] and Schrijver's book [2].

Definition 2.1. A **polyhedron** $P \subset \mathbb{R}^d$ is the intersection of a finite number of closed half-spaces. When the intersection is bounded, we call P a polytope.

A collection of half-spaces can describe any convex polytope. Still, since the constraints are linear, it can also be specified as its vertices' convex hull (the collection of all convex combinations of the vertices). The former is known as the H-description of P, and the latter the V-description of P.

Definition 2.2. A **face** of a polytope P is the intersection of P with a closed half-space whose boundary is disjoint from the interior of P.

We called F a k-face of P when $\dim(F) = k$. The set of 0-faces are the vertices of P, and the 1-faces are the edges of P.

Lemma 2.1 (Farkas' lemma)**.** *Given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. The exactly one of the following statements is true:*

- *$\exists x \in \mathbb{R}^n$ such that $Ax = b$ and $x \geq 0$.*
- *$\exists y \in \mathbb{R}^m$ such that $A^T y \geq 0$ and $b^T y < 0$.*

Multiple theorems exposed throughout the text hold due to Farkas' lemma. This lemma can be interpreted geometrically. The set $\{Ax : x \geq 0\}$ defines a polyhedron. If b lies inside this object, the first part of Farkas' lemma holds. Otherwise, the second part holds.

Definition 2.3. A **directed graph** (or digraph) $G = (V, E)$ is a set of vertices or nodes V and edges E. The edges are of the form (i, j) and correspond to arrows going from node i to node j.

We denote $\delta^+(v)$ to all the leaving edges of v (i.e. are of the form $e = (v, w)$ for some $w \in G$), and $\delta^-(v)$ to all the incoming edges of v.

Definition 2.4. An **algorithm** is a finite set of instructions that will be performed to an input (initial data). A **polynomial-time** algorithm is an algorithm that halts after less than a number of steps polynomial in the input [1].

With this definition of measuring an algorithm's speed, we can introduce the complexity classes P and NP. P refers to the set of problems that can be solved in polynomial time. NP is formally defined as the class of problems solvable by non-deterministic Turing machines. But we will instead think of them as problems that can either be solved in polynomial or exponential time in our computers. An essential subset of NP problems is NP-complete problems. Any NP problem can be reduced to an NP-complete problem. Thus if one proves that an NP-complete problem can be solved in polynomial time, every NP problem will be polynomially solvable.

Definition 2.5. A polytope P is a **transportation polytope** if there exist two non-negative integer vectors $a = (a_1, \ldots, a_n)$, $b = (b_1, \ldots, b_m)$ such that $\sum_{i=1}^n a_i = \sum_{i=1}^m b_i$, and P satisfies the following linear inequalities

$$\begin{aligned}
\sum_{i=1}^{n} x_{i,j} &= b_j \quad \text{for } 1 \leq i \leq m, \\
\sum_{j=1}^{m} x_{i,j} &= a_i \quad \text{for } j \leq i \leq n, \\
x_{i,j} &\geq 0.
\end{aligned}$$

For this case we denote the polytope P by $T(a, b)$.

3 Linear Programming

Convex programming is an important subfield of mathematical optimization that studies optimization on convex sets. A particular type of convex sets are polyhedra, which are convex sets specified by linear constraints. LP is a type of convex programming that aims to find maximal (or minimal) solutions to problems where the objective (or cost) function is linear, and the possible solutions set is defined by linear restrictions. Due to the nature of LP problems, they are of the form

[1] The familiar reader will know that this time is defined with respect to a Turing machine.

$$\max\left\{c^\top x : x \in \mathbb{R}^n, Ax \leq b\right\} \tag{1}$$

where $c^\top \in (\mathbb{R}^n)^*$ is the linear objective function, and $A \in M_{m\times n}(\mathbb{R})$ and $b \in \mathbb{R}^m$ specify the linear constraints that x must satisfy. Geometrically speaking, the solution must lie on or inside the polyhedron with H-description $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Additionally, we can classify the possible problems regarding the associated polyhedron. If the inequalities $Ax \leq b$ cannot be satisfied, the LP problem is **unfeasible**; otherwise, the problem is **feasible**. In the latter case, the polyhedron in consideration can be either bounded or unbounded. We will focus on the bounded case, which means that the solution of the LP problem that we will consider will lie on a polytope. It is a fact that an optimal solution to LP problems will be found on a vertex of the associated polytope. We can view this geometrically since the objective function defines a hyperplane $H_k = \{x \in \mathbb{R}^n : c^\top x = k\}$, thus to maximize k, we should move it along the direction of c (i.e. increasing k) until the intersection of the hyperplane, and the polytope P is non-empty, and if we move more along c, the intersection $P \cap H_k$ is empty. Therefore there are two scenarios. First, the solution is a vertex, and second, the solution is a k-face of P with $1 \leq k$. The former case is referred to as a **non-degenerate** LP problem and the latter as a **degenerate** one.

Likewise, every LP problem (1) has an associated dual LP problem of the form:

$$\min\left\{b^\top y : y \in \mathbb{R}^n, y \geq \mathbf{0}, A^\top y = c^\top\right\} \tag{2}$$

In a sense, it transforms the LP problem's constraints and variables (1) to variables and constraints in the dual (2), respectively.

The following theorem shows the importance of the dual LP problem:

Theorem 3.1 (Strong duality)**.** *Let $A \in M_{m\times n}(\mathbb{R})$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. Then*

$$\max\left\{c^\top x \mid Ax \leq b\right\} = \min\left\{y^\top b \mid y \geq \mathbf{0}, y^\top A = c^\top\right\}$$

if any optimal solution, either $c^\top x$ or $y^\top b$, is finite.

This duality can be useful for writing algorithms that solve a specific LP problem since the dual of a problem could be easier to solve than the original LP problem.

The following example (retrieved from [3]) illustrates the broad applicability of LP:

Example 3.1. A company wants to determine the production rate over the next weeks $[0, T]$ such that the demand $g(t)$ is satisfied and the inventory cost $y(t)$ and production cost $x(t)$ are minimized. If the initial inventory is $y(0) = y_0$ and the desired inventory at the end of the weeks $y(T) = y_T$. Supposing a linear cost for the inventory c_1 and the production c_2, and a bound over the inventory capacity b_1 and production capacity b_2, the total cost we wish to minimize is given by

$$\min\left\{\int_0^T \left[c_1 y(t) + c_2 x(t)\right] dt\right\}$$

with constraints

$$\begin{aligned} &y(t) = y_0 + \textstyle\int_0^t [x(\tau) - g(\tau)] d\tau \\ &y(T) = y_T \\ &0 \leq x(t) \leq b_1 \\ &0 \leq y(t) \leq b_2, \quad t \in [0, T] \end{aligned}$$

we can approximate this complicated minimization problem to an LP problem (2) with arbitrary precision $\varepsilon = \frac{T}{n}$ of the form

$$\begin{aligned} \min \quad & \textstyle\sum_{j=1}^n (c_1\varepsilon)\, y_j + \sum_{j=1}^n (c_2\varepsilon)\, x_j \\ \text{s.t.} \quad & y_j = y_{j-1} + (x_j - g_j)\,\varepsilon \\ & y_n = y_T \\ & 0 \leq x_j \leq b_1 \\ & 0 \leq y_j \leq b_2 \end{aligned}$$

We will focus on a particular type of LP problems called integer programming (ILP) problems. The difference with general LP, as the name suggests, is that the matrix A, the vector b, and c of (1) are elements of $\mathbb{Z}^{m\times n}$, $\mathbb{Z}^m$, and $\mathbb{Z}^n$, respectively. Even though this problem seems easier than general LP problem, it is much harder (NP-complete) since the solutions we are now looking for are also required to be integer $x \in \mathbb{Z}^n$. The general form of an ILP is the following:

$$\begin{aligned} \max \quad & c^{\mathrm{T}}x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq \mathbf{0} \\ & x \in \mathbb{Z}^n \end{aligned}$$

with A, b, and C integer valued. The following result on integer programming [4] will be relevant for the next sections.

Theorem 3.2 (Hoffman (1974), Edmonds and Giles (1977))**.** *If P is a rational polytope (the vertices have rational coordinates), then the following statements are equivalent*

- *P is integral (the vertices have integer coordinates).*
- *Each face of P have integer vectors.*
- *For each c integral, $\max\{cx : x \in P\}$ is obtained by an integer vector x when the maximum is finite.*
- *For each c integral, $\max\{cx : x \in P\}$ is integer when the maximum is finite.*

4 Network flow

Some network flow theory problems can be formulated as LP problems. We will introduce some essential concepts of network flow in this section and elaborate more in Section 7. Let V be a directed graph and E its set of edges, each (i, j) with an associated capacity u_{ij}. A flow $f : E \to \mathbb{R}_+$ in the graph is defined as a set of flows $f(e) = f_{ij} \in \mathbb{R}$ for each edge $e = (i, j)$ such that $f(e) = f((i, j)) \leq u_{ij}$. We define the **excess** of a flow f by

$$\mathrm{ex}_f(v) := \sum_{e\in\delta^-(v)} f(e) - \sum_{e\in\delta^+(v)} f(e).$$

If $\mathrm{ex}_f(v) = 0$, we say there is a flow conservation at v. Let s be the source and t the sink of a directed graph, meaning that there are just outgoing edges on s and incoming edges on t. Then an **s-t flow** is a flow f on G such that $\mathrm{ex}_f(s) \leq 0$ and $\mathrm{ex}_f(v) = 0$ for all $v \in G\backslash\{s, t\}$. The **value** of a flow f is defined by $\mathrm{Val}(f) := -\mathrm{ex}_f(s)$. The problem we are interested in solving is the following: Given a directed graph with capacity u, source s, and sink t. Find

$$\max_{f \text{ flow of } G}\{\mathrm{Val}(f)\}.$$

This problem can be formulated as a LP problem as:

$$\begin{aligned} \max \quad & \sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e) \\ \text{s.t.} \quad & \sum_{e\in\delta^-(v)} f(e) = \sum_{e\in\delta^+(v)} f(e), \quad v \in G\backslash\{s, t\} \\ & f(e) \leq u(e), \quad e \in E \\ & f(e) \geq 0. \end{aligned}$$

We can generalize **s-t flow** to **minimum cost flow** by introducing multiple sources and sinks, and by including some cost in each edge of the graph.

Definition 4.1. Given $G = (V, E)$ be a directed graph with capacities $u_e \geq 0$ and costs $c_e \in \mathbb{R}$ for each $e \in E$. Let $b(v) \in \mathbb{R}$ be a set of numbers (known as balance) for each vertex $v \in V$ such that $\sum_{v \in V(G)} b(v) = 0$. A **b-flow** is a flow $f : E \to \mathbb{R}_+$ such that $f(e) \leq u_e$ for all edges and $\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v)$ for all vertices.

The minimum cost flow problem is to find a b-flow f whose cost

$$c(f) := \sum_{e \in E(G)} f(e) c_e$$

is minimum (or non-existing). We illustrate this concept in the following example:

Example 4.1. Consider the digraph $G = (V, E)$ with $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 2), (3, 4), (4, 1)\}$. The costs for the edges are:

$$\begin{aligned} &c_{12} = 2 \quad c_{13} = -5, \quad c_{23} = -1, \\ &c_{24} = 4, \quad c_{32} = 6, \quad c_{34} = 3, \\ &c_{41} = 7. \end{aligned}$$

The balance for the edges are: $b(1) = 4$, $b(2) = 2$, $b(3) = -1$, $b(4) = -5$. We represent the minimum cost flow problem by the diagram in Figure 1. For this case, the linear constraints are

$$\begin{aligned} f_{12} + f_{13} - f_{41} &= 4 \\ f_{23} + f_{24} - f_{32} - f_{12} &= 2 \\ f_{32} + f_{34} - f_{13} - f_{23} &= -1 \\ f_{41} - f_{24} - f_{34} &= -5 \end{aligned}$$

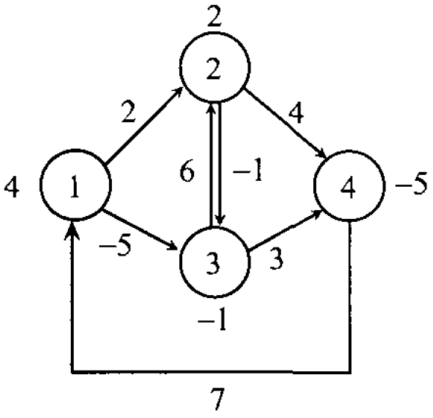

Figure 1: A network flow graph with labels inside the vertices. Each edge has an associated cost next to it, and each node has a balance $b(v)$. Modified from [3].

A particular minimum cost flow problem will be studied and solved in Section 7.

5 Algorithms

There are multiple algorithms to solve an LP problem (1) determined by A, b, and c. The first proposed algorithm to solve these problems is called the Simplex method.

Denote J a set of row indices and A_J the matrix consisting of the J rows of A, b_J the vector consisting of the J components of b. Given $A \in M_{m\times n}(\mathbb{R})$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and a vertex x of $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. The procedure of this algorithm is the following [4]:

1. Choose J of size n such that $\det(A_J) \neq 0$ and $A_J x = b_J$.
2. Set $y_J = c(A_J)^{-1}$ and let $yA = c$ be the extension of y_J by appending zeros in the remaining components. If $y \geq 0$ **halt** and return x.
3. Choose the smallest $i \in \{1, ..., m\}$ that satisfies $y_i < 0$. Let w be the column of $(A_J)^{-1}$. If $Aw \geq 0$ then **halt** and return $-w$.
4. Let $\lambda := \min\left\{\frac{b_{\{j\}} - A_{\{j\}}x}{A_{\{j\}}w} : j \in \{1, \ldots, m\}, a_j w > 0\right\}$ and j the smallest index (if multiple) with this minimum.
5. Let $J = (J\backslash\{i\}) \cup \{j\}$ and $x = x + \lambda w$. Go to step 2.

Theorem 5.1 (Dantzig)**.** *The simplex algorithm takes at most* $\binom{m}{n}$ *iterations to terminate. If the algorithm halts in* ***step 2****, the problem is bounded. If it halts in* ***step 3****, the problem is unbounded.*

There is a variant for the simplex algorithm called the revised simplex algorithm, which optimizes the simplex algorithm by avoiding unnecessary calculations, thereby being faster. Another important family of algorithms is interior-point methods. These methods come from nonlinear programming. While the simplex method searches for solutions on the feasible region's boundary doing many computationally inexpensive iterations, the interior-point methods search for optimal solutions outside or inside the feasible region. The interior-point (IP) methods do fewer iterations, but these are computationally more expensive than the simplex method.

We give a brief overview of a well known IP method called the Primer-dual IP method. Given an LP problem (1), we can define its dual (2). From the dual, we introduce two variables s and μ such that the problem we seek to solve is:

$$\begin{array}{ll} \min & b^T y \\ \text{s.t.} & A^T y - s = c \\ & s \geq 0 \end{array}$$

An equivalent formulation for such a problem is done by including the restrictions in the minimizing function. The constraints over s can be included in terms of a logarithmic barrier function[2] and the equality $A^T y - s = c$ with Lagrange multipliers λ.

$$\min\{F(s, y, \lambda) = b^T y - \mu \sum_{i=1}^{m} \log(s_i) - \lambda^T (A^T y - s)\}$$

We can now impose that the derivatives with respect to all variables must vanish at a minimum, and use Newton's method to update the values of y, s and λ. Due to the convexity of F, Newton's method will converge with arbitrary precision.

6 Birkhoff polytope

Definition 6.1. A matrix $X \in M_{m\times n}(\mathbb{R})$ is **doubly stochastic** if

- Every entry of X is non-negative, $x_{ij} \geq 0$.
- The sum of every row and every column of X is equal to 1, i.e. $\sum_{i=1}^{n} x_{ij} = 1$ and $\sum_{j=1}^{n} x_{ij} = 1$ for all i, j .

[2] A function that takes huge values when the constraints are not satisfied

Theorem 6.1 (Birkhoff - von Neumann)**.** *Any doubly stochastic matrix M is a convex combination of the permutation matrices. We will denote the convex hull of the $n \times n$ permutation matrices, also known as the* ***Birkhoff polytope****, by Ω_n.*

Notice that Ω_n is equal to $T(\mathbb{1}, \mathbb{1})$ where $\mathbb{1} = (1, \ldots, 1) \in \mathbb{R}^n$. Furthermore, from Theorem 6.1, we can see that Ω_n has $n!$ vertices, each corresponding to a permutation matrix. The Birkhoff Polytope is by itself an interesting geometrical object which is not fully understood. Some pure mathematical problems of this polytope, such as calculating its volume or Erhart polynomial for arbitrary n, remain unanswered. Other practical issues, such as the expected run time of algorithms on it, like the simplex method, have been studied in [5]. The authors prove a theorem that tells us how good a modified and simpler version of the simplex method runs in the Birkhoff polytope. This simpler version consists of starting at some vertex and moving to an adjacent vertex along an edge on which the objective function decreases. They define the linear function as

$$\phi = x_{1,1} + \alpha x_{1,2} + \cdots + \alpha^{n-1} x_{1,n} + \alpha^n x_{2,1} + \cdots + \alpha^{n^2-1} x_{n,n}, \tag{3}$$

and prove that for $\alpha \leq \alpha_0 = 1/(n+1)$, this function has a worst case scenario for the algorithm with exponential running time but still find that the expected running time is not exponential.

Theorem 6.2 ([5] Theorem 1.5)**.** *Let ϕ be as in (3), $0 \leq \alpha \leq \alpha_0$. Then, the expected running time E of the algorithm is $E = O(n \log n)$*

Besides, there is no explicit formula for the Birkhoff polytope volume, nor for its faces, from a combinatorial perspective. However, some proposes have been made [6]. Nevertheless, there is a complex-analytic way of calculating the volume [7], which still is hugely costly computationally. In 2003, Matthias Beck and Dennis Pixton found (after a year of calculations with multiple computers) that the volume for Ω_n for $n = 10$ is

$$\frac{7272912840167864209775084579901218625488232600525573333866078 89}{828160860106766855125676318796872729344622463533089422677980721388055739956270293750883504892820848640000000}$$

This is the largest n for which the volume of Ω_n has been calculated. Despite the unusual volume of such object, we will see that this polytope appears in a relatively common problem in the next section.

7 The Assignment problem

Consider the following scenario: we want to transport a certain amount of a 'commodity' from m starting points to n destination points. Each starting point i has a supply of s_i commodity units, and each destination j requires d_j units. Also, for each pair (i, j) between a starting and ending point, there is a cost c_{ij} per transported unit. Thus, we want to find the best distribution of commodity units possible, i.e. the one minimizing the total transportation cost.
Let x_{ij} be the number of units shipped from node i to node j. The problem is **balanced** if

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j.$$

When the total supply exceeds the total demand, we can add a destination $d+1$ having demand $d_{n+1} = \sum_{i=1}^{m} s_i - \sum_{j=1}^{n} d_j$ and $c_{i,n+1} = 0$ for $i = 1, \ldots, m$ to obtain a balanced problem. In this case, the objective function which we want to minimize is

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

with linear constraints

$$\sum_{j=1}^{n} x_{ij} = s_i \qquad \text{for } i = 1, \ldots, m,$$

$$\sum_{i=1}^{m} x_{ij} = d_j \qquad \text{for } j = 1, \ldots, n$$

$$x_{ij} \geq 0 \qquad \text{for } i = 1, \ldots, m \text{ and } j = 1, \ldots, n$$

This problem is a particular case of a b-flow, and we illustrate it in Figure 2.

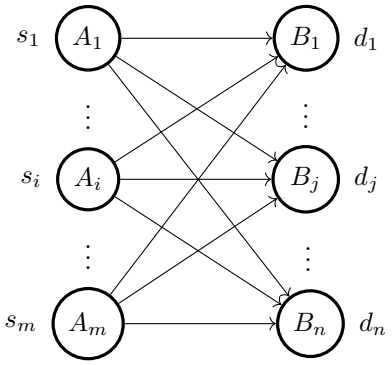

Figure 2: General graph representation of a transportation problem

An important and special case of the transportation problem is when $m = n$, and $s_i = 1$ and $d_j = 1$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. This instance is called the assignment problem. The LP formulation for this Network-flow problem is given by

$$\min \left\{ \sum_{i,j=1}^{n} c_{ij} x_{ij} : \forall i, j \in [n] \quad \sum_{i=1}^{n} x_{ij} = 1, \sum_{j=1}^{n} a_{ij} = 1, x_{ij} \in \{0, 1\} \right\} \tag{4}$$

and its dual by

$$\max \left\{ \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j : \forall i, j \in [n] \quad \sum_{i=1}^{n} u_i + v_j \leq c_{ij}, u_i \in \mathbb{R}, v_j \in \mathbb{R} \right\}.$$

The feasible region of (4) is the set of doubly stochastic matrices. Due to the Birkhoff theorem, the solution of (4) will be a permutation matrix.

We consider the following example of an assignment problem (4): suppose that we have m individuals in a building with k floors and m elevators. We want to pair the people with the elevators such that the sum of the distance travelled by the elevators is minimum. In other words, we want to find the best energy-efficient way of distributing the elevators around the building. In the form of an assignment problem, the cost c_{ij} is the distance between the person i and the elevator j. We implement different LP algorithms for solving this example. The code can be found in the following repository:https://github.com/BestQuark/Elevator-Polytopes .

8 Conclusion

Linear programming is a vast and well-studied mathematical field of optimization. The benefits of understanding the geometry of the problems in question arise when finding fast algorithms or when trying to improve an old one. For example, the core of the simplex algorithm is representing an LP problem as a convex polytope since it moves along the boundary of it. Some specific LP problems, such as the assignment problem, have a rich mathematical structure, where the geometrical objects in consideration are still under study, as for the Birkhoff Polytope.

In this text, we reviewed the core notions of linear programming for studying the assignment problem. We applied them to a simple elevator system. The tools of LP allows finding an energy-efficient method to distribute the elevators in a building. Moreover, the LP formulation of the elevator system studied can be extended to a more realistic scenario by adding that the elevator not only picks one person at the time but can pick people on the way. In our problem, we did not include if the person is calling the elevator up or down, neither we allow more than one person per elevator. The motivation for the applicability of this example is the use of elevators in our current sanitary situation. It is safer if just one person is inside the elevator to respect social distancing. Likewise, assignment problems appear in many more cases. We could think of an Uber-like app, where some users need a ride and must be assigned to free cars. We could treat this system as an assignment problem where the cost between each node (person and car) is the number of blocks apart. It is up to our imagination to solve many problems with linear programming tools and up to our agility to solve them at the right time.

References

[1] Günter M Ziegler. "Lectures on Polytopes." In: (1993).

[2] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media, 2003.

[3] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[4] Korte Bernhard and Jens Vygen. "Combinatorial optimization: Theory and algorithms". In: *Springer, Sixth Edition, 2018.* (2018).

[5] Igor Pak. "Four questions on Birkhoff polytope". In: *Annals of Combinatorics* 4.1 (2000), pp. 83–90.

[6] Carolina Benedetti et al. "A combinatorial model for computing volumes of flow polytopes". In: *Transactions of the American Mathematical Society* 372.5 (2019), pp. 3369–3404.

[7] Matthias Beck and Dennis Pixton. "The Ehrhart polynomial of the Birkhoff polytope". In: *Discrete & Computational Geometry* 30.4 (2003), pp. 623–637.

[8] Maria Mehlum. "Doubly stochastic matrices and the assignment problem". In: (2012).

[9] Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: theory of majorization and its applications*. Vol. 143. Springer, 1979.

[10] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[11] Farid Alizadeh. "Interior point methods in semidefinite programming with applications to combinatorial optimization". In: *SIAM journal on Optimization* 5.1 (1995), pp. 13–51.