10. LINEAR PROGRAMMING

Serge Gaspers
serge.gaspers@unsw.edu.au

Katie Clinch
k.clinch@unsw.edu.au

Course admin: Song Fang
cs3121@cse.unsw.edu.au

School of Computer Science and Engineering, UNSW Sydney

Term 3, 2023

Table of Contents

What is linear programming?

- It is a method of solving problems where you have a system of linear constraints.

What is a linear constraint?

- A linear constraint is an equality ($=$) or inequality ($\geq$ or $\leq$) where each term has just one unknown value x_i.
- Each unknown value appears as x_i (not $\sqrt{x_i}$ or x_i^3 or $\frac{1}{x_i}$ etc.)
- Each unknown value x_i can be multiplied by any known values.

Example. Suppose you want to buy some paint. The paint cans come in three sizes small v_1, medium v_2 and large v_3. A can of size v_i has cost c_i. You want to decide how many cans n_i to buy of each size v_i.

- **What are the unknown values?** n_1, n_2, n_3.
- **What are the known values?** $v_1, v_2, v_3, c_1, c_2, c_3$.

Example. Suppose you want to buy some paint. The paint cans come in three sizes small v_1, medium v_2 and large v_3. A can of size v_i has cost c_i. You want to decide how many cans n_i to buy of each size v_i.

- **Are the following constraints linear?**
  - $n_1 \geq 0$. Yes
  - $3n_1 + 2n_2 + 10 = 7$. Yes. It would be simpler to write this as $3n_1 + 2n_2 = -3$.
  - $c_1 n_1 - c_2 n_2 + 2c_3 n_3 \leq 100$. Yes
  - $v_1 n_1^2 + v_2 n_2 n_3 \geq 50$. No, n_1^2 and $n_2 n_3$ are not linear.
  - $\sum_{i=1}^{3} v_i n_i \geq 60$. Yes
  - $\sum_{i=1}^{3} v_i n_i^2 \geq 50$. No, x_i^2 is not linear.
  - **Hard.** $\sum_{i=1}^{3} c_i v_i n_i \leq 10$. Yes! Each unknown n_i is multiplied by the known value $c_i v_i$.

Problem

Instance: a list of food sources $F_1, \ldots, F_n$; and for each source F_i:

- its price per gram p_i;
- the number of calories c_i per gram, and
- for each of 13 vitamins $V_1, \ldots, V_{13}$, the content $v_{i,j}$ in milligrams of vitamin V_j in one gram of food source f_i.

Task: find a combination of quantities of food sources such that:

- the total number of calories in all of the chosen food is equal to a recommended daily value of 2000 calories;
- for each $1 \leq j \leq 13$, the total intake of vitamin V_j is at least the recommended daily intake of w_j milligrams, and
- the price of all food per day is as low as possible.

Suppose we take x_i grams of each food source F_i for $1 \leq i \leq n$. Then the constraints are as follows.

- The total number of calories must satisfy

$$\sum_{i=1}^{n} c_i x_i = 2000;$$

- For each $1 \leq j \leq 13$, the total amount of vitamin V_j in all food must satisfy

$$\sum_{i=1}^{n} v_{i,j} x_i \geq w_j.$$

- Implicitly, all the quantities must be non-negative numbers, i.e. $x_i \geq 0$ for all $1 \leq i \leq n$.

- Our goal is to minimise the objective function, which is the total cost
$$y = \sum_{i=1}^{n} p_i x_i.$$

- Note that all constraints and the objective function are **linear**.

Linear Programming problems: Example 2

Problem

Instance: you are a politician and you want to ensure an election victory by making certain promises to the electorate. You can promise to build:

- bridges, each costing 3 billion;
- rural airports, each costing 2 billion, and
- Olympic swimming pools, each costing 1 billion.

Problem (continued)

You were told by your wise advisers that

- each bridge you promise brings you 5% of city votes, 7% of suburban votes and 9% of rural votes;
- each rural airport you promise brings you no city votes, 2% of suburban votes and 15% of rural votes;
- each Olympic swimming pool promised brings you 12% of city votes, 3% of suburban votes and no rural votes.

Problem (continued)

In order to win, you have to get at least 51% of each of the city, suburban and rural votes.

Task: decide how many bridges, airports and pools to promise in order to guarantee an election win at minimum cost to the budget.

- Let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$
\begin{aligned}
0.05x_b \phantom{{}+0.02x_a} + 0.12x_p &\geq 0.51 && \text{(city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 && \text{(suburban votes)}\\
0.09x_b + 0.15x_a \phantom{{}+0.03x_p} &\geq 0.51 && \text{(rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}
$$

- However, there is a very significant difference with the first example:
  - you can eat 1.56 grams of chocolate, but
  - you cannot promise to build 1.56 bridges, 2.83 airports and 0.57 swimming pools!
- The second example is an example of an **Integer Linear Programming problem**, which requires all the solutions to be integers.
- Such problems are MUCH harder to solve than the "plain" Linear Programming problems whose solutions can be real numbers.

- We won't see algorithms which solve LP problems in this lecture; we will only study the structure of these problems further.

- There are polynomial time algorithms for Linear Programming, including the ellipsoid algorithm.

- In practice we typically use the SIMPLEX algorithm instead; its worst case time complexity is exponential, but it is very efficient in the 'average' case.

- There is no known polynomial time algorithm for <u>Integer</u> Linear Programming!

How do you solve a (continuous) linear programming problem?

In practice, the steps you will go through are:

1. Convert the requirements of the problem into equalities and inequalities.
2. Check that these are indeed linear constraints, (and that the problem is continuous).
3. If so, input the constraints into an existing LP solver, and get the solution!

So to solve a (continuous) LP problem, you do not need to design a new algorithm. You can just use an existing solver as a black box.

Table of Contents

In the **standard form** the objective to be maximised is given by

$$\sum_{i=1}^{n} c_i \, x_i$$

and the constraints are of the form

$$\sum_{i=1}^{n} a_{ij} x_i \leq b_j \qquad (1 \leq j \leq m);$$

$$x_i \geq 0 \qquad (1 \leq i \leq n).$$

- To get a more compact representation of linear programs, we use vectors and matrices.
- Let $\mathbf{x}$ represent a (column) vector,

$$\mathbf{x} = \langle x_1 \ldots x_n \rangle^T.$$

- Define a partial ordering on the vectors in $\mathbb{R}^n$ by $\mathbf{x} \leq \mathbf{y}$ if and only if the corresponding inequalities hold coordinate-wise, i.e., if and only if $x_i \leq y_i$ for all $1 \leq i \leq n$.

Write the coefficients in the objective function as

$$\mathbf{c} = \langle c_1 \ldots c_n \rangle^T \in \mathbb{R}^n,$$

the coefficients in the constraints as an $m \times n$ matrix

$$A = (a_{ij})$$

and the right-hand side values of the constraints as

$$\mathbf{b} = \langle b_1 \ldots b_m \rangle^T \in \mathbb{R}^m.$$

Then the standard form can be formulated simply as:

- maximize $\mathbf{c}^T\mathbf{x}$
- subject to the following two (matrix-vector) constraints:

$$\begin{aligned} A\mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0}. \end{aligned}$$

Thus, a Linear Programming optimisation problem can be specified as a triplet $(A, \mathbf{b}, \mathbf{c})$, which is the form accepted by most standard LP solvers.

- The Standard Form doesn't immediately appear to handle the full generality of LP problems.

- LP problems could have:
  - equality constraints
  - unconstrained variables (i.e. potentially negative values x_i)
  - absolute value constraints

How do we handle each of these issues?
We will see that there is a way to write each of these three requirements in Standard Form.

Equality constraints

- An LP problem may include equality constraints of the form

$$\sum_{i=1}^{n} a_{ij} x_i = b_j.$$

- Each of these can be replaced by two inequalities:

$$\sum_{i=1}^{n} a_{ij} x_i \geq b_j$$
$$\sum_{i=1}^{n} a_{ij} x_i \leq b_j.$$

- Thus, we can assume that all constraints are inequalities.

- In general, a "natural formulation" of a problem as a Linear Program does not necessarily require that all variables be non-negative.

- However, the Standard Form does impose this constraint.

- This poses no problem, because each occurrence of an unconstrained variable x_i can be replaced by the expression

$$x_i' - x_i^*$$

where x_i', x_i^* are new variables satisfying the inequality constraints

$$x_i' \geq 0, \ x_i^* \geq 0.$$

Absolute value constraints

- For a vector

  $$\mathbf{x} = \langle x_1, \ldots, x_n \rangle^T,$$

  we can define

  $$|\mathbf{x}| = \langle |x_1|, \ldots, |x_n| \rangle^T.$$

- Some problems are naturally translated into constraints of the form

  $$|A\mathbf{x}| \leq \mathbf{b}.$$

- This also poses no problem because we can replace such constraints with two linear constraints:

  $$A\mathbf{x} \leq \mathbf{b} \text{ and } -A\mathbf{x} \leq \mathbf{b},$$

  because $|x| \leq y$ if and only if $x \leq y$ and $-x \leq y$.

Standard Form

maximize

$$\mathbf{c}^T\mathbf{x}$$

subject to

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

and

$$\mathbf{x} \geq \mathbf{0}.$$

Any vector $\mathbf{x}$ which satisfies the two constraints is called a feasible solution, regardless of what the corresponding objective value $\mathbf{c}^T\mathbf{x}$ might be.

A vector $\mathbf{x}$ that satisfies all three constraints is called an optimal solution.

Table of Contents

As an example, let us consider the following optimisation problem.

Problem

$$\begin{aligned} \text{maximise} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (1) \\ \text{subject to} \quad & && \\ & x_1 + x_2 + 3x_3 \leq 30 && (2) \\ & 2x_1 + 2x_2 + 5x_3 \leq 24 && (3) \\ & 4x_1 + x_2 + 2x_3 \leq 36 && (4) \\ & x_1, x_2, x_3 \geq 0 && (5) \end{aligned}$$

Question

How large can the value of the objective

$$z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3$$

be, without violating the constraints?

We can achieve a crude bound by adding inequalities (2) and (3), to obtain

$$3x_1 + 3x_2 + 8x_3 \leq 54.$$

Since all variables are constrained to be non-negative, we are assured that

$$3x_1 + x_2 + 2x_3 \leq 3x_1 + 3x_2 + 8x_3 \leq 54,$$

i.e. the objective does not exceed 54. Can we do better?

We could try to look for coefficients $y_1, y_2, y_3 \geq 0$ to be used to form a linear combination of the constraints:

$$y_1(x_1 + x_2 + 3x_3) \leq 30y_1 \tag{6}$$
$$y_2(2x_1 + 2x_2 + 5x_3) \leq 24y_2 \tag{7}$$
$$y_3(4x_1 + x_2 + 2x_3) \leq 36y_3 \tag{8}$$

Then, summing up all these inequalities and factoring, we get

$$\begin{aligned} & x_1(y_1 + 2y_2 + 4y_3) \\ & + x_2(y_1 + 2y_2 + y_3) \\ & + x_3(3y_1 + 5y_2 + 2y_3) \\ & \leq 30y_1 + 24y_2 + 36y_3. \end{aligned}$$

If we compare this with our objective (1) we see that if we choose y_1, y_2 and y_3 so that:

$$\begin{aligned} y_1 + 2y_2 + 4y_3 &\geq 3 \\ y_1 + 2y_2 + y_3 &\geq 1 \\ 3y_1 + 5y_2 + 2y_3 &\geq 2 \end{aligned}$$

then

$$\begin{aligned} 3x_1 + x_2 + 2x_3 \leq\; & x_1(y_1 + 2y_2 + 4y_3) \\ & + x_2(y_1 + 2y_2 + y_3) \\ & + x_3(3y_1 + 5y_2 + 2y_3). \end{aligned}$$

Combining this with (6) – (8) we get

$$30y_1 + 24y_2 + 36y_3 \geq 3x_1 + x_2 + 2x_3 = z(x_1, x_2, x_3).$$

Consequently, in order to find a tight upper bound for our objective $z(x_1, x_2, x_3)$ in the original problem P, we have to find y_1, y_2, y_3 which solve problem P^*:

$$\text{minimise:} \quad z^*(y_1, y_2, y_3) = 30y_1 + 24y_2 + 36y_3 \tag{9}$$

subject to:

$$y_1 + 2y_2 + 4y_3 \geq 3 \tag{10}$$

$$y_1 + 2y_2 + y_3 \geq 1 \tag{11}$$

$$3y_1 + 5y_2 + 2y_3 \geq 2 \tag{12}$$

$$y_1, y_2, y_3 \geq 0 \tag{13}$$

Then

$$\begin{aligned} z^*(y_1, y_2, y_3) &= 30y_1 + 24y_2 + 36y_3 \\ &\geq 3x_1 + x_2 + 2x_3 \\ &= z(x_1, x_2, x_3) \end{aligned}$$

will be a tight upper bound.

The new problem P^* is called the <u>dual problem</u> of P.

Let us now repeat the whole procedure in order to find the dual of P^*, which will be denoted $(P^*)^*$.
We are now looking for $z_1, z_2, z_3 \geq 0$ to multiply inequalities (10)–(12) and obtain

$$\begin{aligned} z_1(y_1 + 2y_2 + 4y_3) &\geq 3z_1 \\ z_2(y_1 + 2y_2 + y_3) &\geq z_2 \\ z_3(3y_1 + 5y_2 + 2y_3) &\geq 2z_3 \end{aligned}$$

Summing these up and factoring produces

$$\begin{aligned} & y_1(z_1 + z_2 + 3z_3) \\ & + y_2(2z_1 + 2z_2 + 5z_3) \\ & + y_3(4z_1 + z_2 + 2z_3) \\ & \geq 3z_1 + z_2 + 2z_3 \end{aligned} \tag{14}$$

If we choose multipliers z_1, z_2, z_3 so that

$$\begin{aligned} z_1 + z_2 + 3z_3 &\leq 30 \\ 2z_1 + 2z_2 + 5z_3 &\leq 24 \\ 4z_1 + z_2 + 2z_3 &\leq 36 \end{aligned}$$

we will have:

$$\begin{aligned} & y_1(z_1 + z_2 + 3z_3) \\ & + y_2(2z_1 + 2z_2 + 5z_3) \\ & + y_3(4z_1 + z_2 + 2z_3) \\ & \leq 30y_1 + 24y_2 + 36y_3 \end{aligned}$$

Combining this with (14) we get

$$3z_1 + z_2 + 2z_3 \leq 30y_1 + 24y_2 + 36y_3.$$

Consequently, finding the double dual program $(P^*)^*$ amounts to maximising the objective $3z_1 + z_2 + 2z_3$ subject to the constraints

$$\begin{aligned} z_1 + z_2 + 3z_3 &\leq 30 \\ 2z_1 + 2z_2 + 5z_3 &\leq 24 \\ 4z_1 + z_2 + 2z_3 &\leq 36 \\ z_1, z_2, z_3 &\geq 0 \end{aligned}$$

This is exactly our starting program P, with only the variable names changed! Thus, the double dual program $(P^*)^*$ is just P itself.

But why is this helpful?

- It appeared at first that looking for the multipliers y_1, y_2, y_3 did not help much, because it only reduced a maximisation problem to an equally hard minimisation problem.
- It is useful at this point to remember how we proved that the Ford-Fulkerson algorithm produces a **maximal flow**, by showing that it terminates only when we reach the capacity of a **minimal cut**.

Why is the dual helpful?

- Finding a feasible solution to P is “easy”, but finding an optimal solution may be hard.
- Similarly finding a feasible solution to P^* is “easy”.
- For any feasible solutions $\mathbf{x}$ of P and $\mathbf{y}$ of P^* we have

$$z(\mathbf{x}) \leq z^*(\mathbf{y})$$

- When these two values are close, then $\mathbf{x}$ is close to the optimal solution of P.
- By cleverly searching for feasible solutions of both the primal P and the dual P^*, we can get:
  - fast approximation algorithms for P,
  - more efficient exact algorithms for P.
- We’ll explain these ideas more formally in the next section.

Table of Contents

In general, the primal Linear Program P and its dual P^* are:

$$P: \quad \text{maximize} \quad z(\mathbf{x}) = \sum_{i=1}^{n} c_i x_i,$$

$$\text{subject to} \quad \sum_{i=1}^{n} a_{ij} x_i \leq b_j \qquad (1 \leq j \leq m)$$

$$\text{and} \quad x_1, \ldots, x_n \geq 0;$$

$$P^*: \quad \text{minimize} \quad z^*(\mathbf{y}) = \sum_{j=1}^{m} b_j y_j,$$

$$\text{subject to} \quad \sum_{j=1}^{m} a_{ij} y_j \geq c_i \qquad (1 \leq i \leq n)$$

$$\text{and} \quad y_1, \ldots, y_m \geq 0.$$

Primal and dual linear programs

We can equivalently write P and P^* in matrix form:

Primal

$$\begin{aligned} P:\ \ & \text{maximize} & z(\mathbf{x}) = \mathbf{c}^T\mathbf{x}, \\ & \text{subject to} & A\mathbf{x} \le \mathbf{b} \\ & \text{and} & \mathbf{x} \ge 0; \end{aligned}$$

Dual

$$\begin{aligned} P^*:\ \ & \text{minimize} & z^*(\mathbf{y}) = \mathbf{b}^T\mathbf{y}, \\ & \text{subject to} & A^T\mathbf{y} \ge \mathbf{c} \\ & \text{and} & \mathbf{y} \ge 0. \end{aligned}$$

Recall that any vector $\mathbf{x}$ which satisfies the two constraints $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$ is called a feasible solution, regardless of what the corresponding objective value $\mathbf{c}^T\mathbf{x}$ might be.

Theorem

If $\mathbf{x} = \langle x_1 \ldots x_n \rangle$ is any feasible solution for P and $\mathbf{y} = \langle y_1 \ldots y_m \rangle$ is any feasible solution for P^*, then:

$$z(\mathbf{x}) = \sum_{i=1}^{n} c_i x_i \leq \sum_{j=1}^{m} b_j y_j = z^*(\mathbf{y})$$

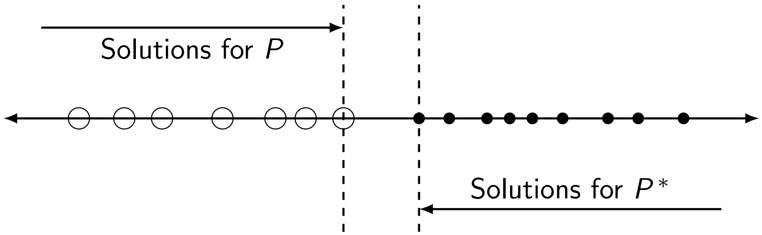
Proof

Since **x** and **y** are feasible solutions for P and P^* respectively, we can use the constraint inequalities for P^* and P that were given on slide 39 to obtain

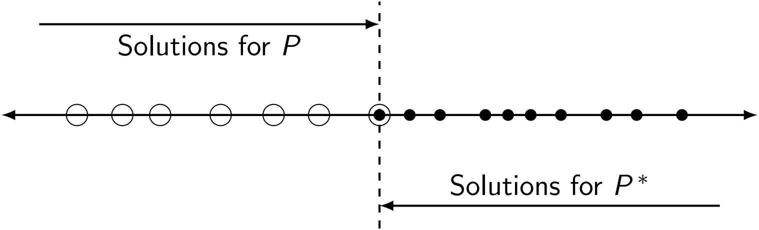
$$\begin{aligned} z(\mathbf{x}) &= \sum_{i=1}^{n} c_i x_i \leq \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ij} y_j \right) x_i \\ &= \sum_{j=1}^{m} \left(\sum_{i=1}^{n} a_{ij} x_i \right) y_j \leq \sum_{j=1}^{m} b_j y_j \\ &= z^*(\mathbf{y}). \end{aligned}$$

Thus, the value of (the objective of P^* for) any feasible solution of P^* is an upper bound for the set of all values of (the objective of P for) all feasible solutions of P.

Similarly, every feasible solution of P is a lower bound for the set of feasible solutions for P^*.



If we find a feasible solution for P which is equal to a feasible solution to P^*, this common value must be the maximal feasible value of the objective of P and the minimal feasible value of the objective of P^*.



- If we use a search procedure to find an optimal solution for P we know when to stop: when such a value is also a feasible solution for P^*.
- This is why the most commonly used LP solving method, the SIMPLEX method, produces an optimal solution for P: because it stops at a value of the primal objective which is also a value of the dual objective.
- See the supplemental notes for the details and an example of how the SIMPLEX algorithm runs.

Table of Contents

Puzzle

There are five sisters in a house.

- Sharon is reading a book.
- Jennifer is playing chess.
- Catherine is cooking.
- Anna is doing laundry.

What is Helen, the fifth sister, doing?