Introduction to Linear Optimization Lecture 1

Martin Böhm

University of Wrocław, Winter 2023/2024

D(Linear program**):** A *linear program* is a model of an optimization problem over real-valued variables using linear constraints and a linear objective function.

- A linear function: $c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3 + \cdots + c_n x_n$. c_i are *constants*, x_i are *variables.*
- A linear constraint: $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \le b$. b is a *right-hand-side constant.*

Typical LP:

$$\begin{array}{llllll} \min & c_1x_1 & +c_2x_2+ & \dots + c_nx_n & \\ \text{s.t.} & a_{1,1}x_1 & +a_{1,2}x_2+ & \dots + a_{1,n}x_n & \ge b_1 \\ & a_{2,1}x_1 & +a_{2,2}x_2+ & \dots + a_{2,n}x_n & \ge b_2 \\ & & & \dots & \\ & a_{m,1}x_1 & +a_{m,2}x_2+ & \dots + a_{m,n}x_n & \ge b_m \\ & & \forall i\colon x_i \ge 0, x_i \in \mathbb{R} & & \end{array}$$

Compact form:

$$\begin{array}{ll} \min & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} \ge \boldsymbol{b} \\ & \boldsymbol{x} \ge 0 \end{array}$$

T: An optimal vector $\boldsymbol{x}^*$ to a linear program can be computed in polynomial time with respect to the program size.

Notes:

- We can use any linear inequalities we want – $\ge, \le$ and $=$.
- We can use maximum instead of minimum.
- If our model has no feasible solution, the solver will inform us about this also in polynomial time.
- We can also detect unboundedness in polynomial time.

Five reasons why I love linear programming

1. Linear programs of polynomial size are efficiently solvable – they have a polynomial bound in the worst-case, and they have very fast practical algorithms (called *solvers*) which are publicly available and significantly faster than the worst-case bounds on most real-world instances.
2. Linear programming proved extremely useful in practice throughout the 20th and 21st century, with a Nobel prize in economics being awarded to the field as well as being a crucial model for planning, routing and scheduling to this day.
   Personal anecdote: While I was in Germany in 2018, my group was actively collaborating with the steel-mill industry where linear programming was used to optimize time and resources.
3. A modeling tool: If you can describe it, it is solvable. In classical Algorithm courses, it is sometimes easy to describe a problem but then its complexity is difficult or even impossible to resolve. Linear programming is P-complete (see below) and thus there is no way to state a hard problem at all. (Of course, a transformation from a problem in P to its linear program may not be obvious.)
4. Every solution comes with a certificate. In other words, once an optimum solution is found, the complexity of verifying its optimality is equivalent to verifying a specific set of linear inequalities, which can be done very efficiently.
5. Linear programming has seen extensive use in approximation algorithms. Approximation algorithms is a field which designs polynomial time algorithms that solve an optimization problem (whose decision version is NP-complete), but return a solution that is at most c-times worse, where c is a small value – common values are 2 or $1 + \varepsilon$.

Linear programming and complexity theory

T: If a linear program is bounded and has at least one feasible solution, it always has an optimal solution.

P: The proof comes later in the semester.

D(Integer linear program**):** An integer linear program is an optimization problem over *integer* variables using linear constraints and a linear objective function.

Compact form:

$$\begin{array}{ll} \min & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} \ge \boldsymbol{b} \\ & \boldsymbol{x} \ge 0, \boldsymbol{x} \in \mathbb{N} \end{array}$$

We also talk about *mixed integer linear programming* (MILP) when some variables are real-valued, and some variables are integer-valued. This is quite common in real-world usage.

T: Consider the decision problem where we get on input a polynomial-sized integer program and the task is to decide whether it has at least one feasible solution. This problem is NP-complete.

Recall the definition of NP-completeness:

D(NP**):** A decision problem belongs to the class NP if it can be *verified* in polynomial time – there exists a polynomial-time algorithm that answers `Yes` when given the decision problem instance and a correct solution, and will answer `No` on any incorrect solution.

D(NP-hard**):** A decision problem π is NP-hard if all problems in NP can be transformed to π using a polynomial time algorithm called a *reduction.* In other words, this problem π is at least as hard as any problem that belongs to NP.

D(NP-complete**):** A decision problem is NP-complete if it simultaneously belongs to NP and is NP-hard.

Linear programming has a similar completeness property:

D(P**):** A decision problem belongs to the class P if it can be decided by an algorithm in polynomial time.

D(L/LOGSPACE**):** A decision problem belongs to the class L if it can be decided by an algorithm whose writable memory is at most logarithmic in the size of the input.

D(P-hard, P-complete**):** A decision problem π is P-hard if all problems in P can be reduced to it using a reduction algorithm that is in L. A problem π is P-complete if it lies in P and is P-hard.

T: Deciding whether a given linear program has at least one feasible solution is P-complete.

Modeling with LPs

There is no *correct* way to model a problem with a linear program. In fact, many problems have several useful LP formulations.

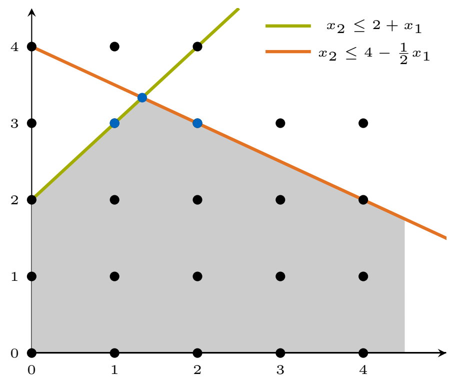
There is no guarantee that a *nice* (readable, reasonable, etc.) model even exists.

Useful questions at the beginning:

1. **Q:** What should our variables $\boldsymbol{x}$ be?
2. **Q:** How should we model the objective function?

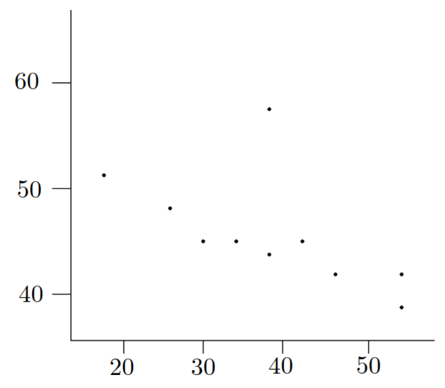
A numerical problem

$$\begin{array}{lll} \max & x_2 & \\ \text{s.t.} & x_2 \le 2 + x_1 & \\ & x_2 \le 4 - \frac{1}{2}x_1 & \\ & x_i \ge 0 & \forall i \in \{1, 2\} \end{array}$$



Line fitting

Task: Given a collection of two-dimensional data points, find a straight line that best approximates the data.



Some possible objectives:

- *Least Squares Method/squared L^2 norm:* $\min \sum_{i=1}^{n}(ax_i + b - y_i)^2$. Computable, but not using linear programming.
- *Minimizing L^1 norm:* $\min \sum_{i=1}^{n} |ax_i + b - y_i|$.

Minimizing the L^1 norm:

$$\begin{aligned}
&\min \sum_{i=1}^{n} e_i \\
\text{s.t. } &\forall i \in \{1,\dots,n\}\colon e_i \geq ax_i + b - y_i \\
&\forall i \in \{1,\dots,n\}\colon e_i \geq -(ax_i + b - y_i) \\
&a \in \mathbb{R}, b \in \mathbb{R}, \forall i\colon e_i \geq 0.
\end{aligned}$$

Flows in networks

Problem Maximum Flow:

Input: An edge-weighted, directed graph $G = (V, E, c)$ (a network). Every directed edge has an associated capacity $c_e \geq 0$. Additionally, there are two special vertices – source s and sink t.

Output: Any flow – function $f : E \to \mathbb{R}_0^+$ which *obeys capacities* ($f(e) \leq c_e$) and *follows Kirchhoff's law* – in every non-special vertex, fluid coming in = fluid coming out.

Goal: Find a flow that sends as much fluid as possible from s to t.

$$\begin{aligned}
&\max \sum_{\vec{si}} x_{\vec{si}} \\
\text{s.t.}\quad &\forall \vec{ij} \in E(G)\colon && x_{\vec{ij}} \leq c_{\vec{ij}} \\
&\forall v \in V(G) \setminus \{s,t\}\colon && \sum_{\vec{vi}} x_{\vec{vi}} - \sum_{\vec{jv}} x_{\vec{jv}} = 0 \\
&\forall \vec{ij} \in E(G)\colon && x_{\vec{ij}} \geq 0, x_{\vec{ij}} \in \mathbb{R}
\end{aligned}$$

T: Maximum Flow is solvable in polynomial time.

Exercises

Exercise one. Formulate the following as a linear program. Use graphical methods (or guesswork) to find the optimal solution.

Ingredients:

	Pizza	Lasagne	available
Tomatoes	2	3	18
Cheese	4	3	24

Profit: Pizza 16 PLN, Lasagne 14 PLN

Task: Determine optimal producible number of pizza and lasagne to maximize total profit.

Exercise two.

Part 1. Suppose we have a system of linear inequalities that also contains sharp inequalities. One that may look like this:

$$\begin{aligned}
5x + 3y &\leq 8 \\
2x - 5z &< -3 \\
6x + 5y + 2w &= 5 \\
3z + 2w &> 5 \\
x, y, z, w &\geq 0
\end{aligned}$$

Is there a way to check if this system has a feasible solution using a linear program?

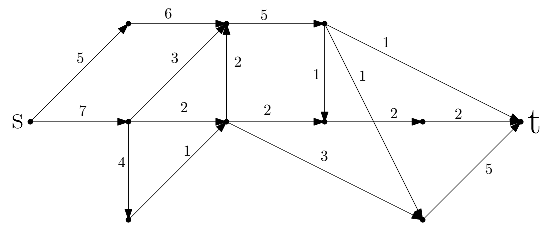
Part 2. Does this mean that linear programming allows strict inequalities? Not really. As a strange example, construct a "linear program with a strict ineqality" that satisfies the following:

- There is a simple finite upper bound on its optimum value;
- There is a feasible solution;
- There is no optimal solution.

This may not happen for a linear program – for a bounded LP, once there exists a feasible solution, there exists also an optimal solution.

Exercise three. Given a graph with directed edges, weights on the edges and two special vertices s and t, we say a *cut* is any collection of edges $S \subseteq E$ whose removal disconnects all directed paths from s to t.

Naturally, since E is a valid cut, we look for the cut with minimum total weight. Using simple logical arguments, can you find the minimum cut on the following directed graph from the lecture?



Exercise four. Design a mixed integer linear program for the problem listed below.

There are n bakeries and m stores in Arbitraryville. Every day the i-th bakery bakes p_i baguettes and the j-th store sells o_j baguettes (we can assume $\sum_{i=1}^{n} p_i = \sum_{j=1}^{m} o_j$). Transporting a baguette from the i-th bakery to the j-th store costs the store c_{ij} crowns.

However! After a few ideal months in Arbitraryville, the stores realized that they need to pay an additional fee l_{ij} for each route from the i-th bakery to the j-th store.

The stores need to pay the fee l_{ij} only when at least a part of a baguette gets transported from the i-th bakery to the j-th store – but the cost is constant and independent of the number of baguettes transported.

Your program should find the optimum flow of baguettes so that each bakery gets rid of all baguettes, each store sells all they can sell and the total fees are minimal.

Exercise five. Suppose we have a real matrix A and appropriate vectors b, c. From those, we can build the following integer program C:

$$\begin{aligned}
&\max c^T x \\
&Ax \leq b \\
&x \in \{0,1\}^n
\end{aligned}$$

Using the same givens, we could also build a linear program L:

$$\begin{aligned}
&\max c^T x \\
&Ax \leq b \\
&x \in [0,1]^n
\end{aligned}$$

Assume that both programs have a solution. Suppose that we pick one optimal solution of the integer program and call it x_C^*, and we also pick one optimal solution of the linear program, denoting it x_L^*. Prove the following inequality:

$$c^T x_C^* \leq c^T x_L^*.$$

Exercise six. Let us consider the following NP-hard problem, called Weighted Vertex Cover:

Input: Undirected graph G with non-negative real weights on the vertices, given by a weight function $w : V(G) \to \mathbb{R}_0^+$.

Goal: To find a subset S of vertices such that each edge $e \in E(G)$ has at least one endpoint in S. (We say that a vertex *covers* an edge, so if we cover all edges, we get a vertex cover.)

From all such subsets S we look for the one with *minimum weight* , i.e. minimum $\sum_{s \in S} w(s)$.

Suggest an integer program with variables $x \in \{0,1\}$ that finds the optimal solution of Weighted Vertex Cover.

Exercise seven. Let us now make use of the previous two exercises and figure out a polynomial-time algorithm that finds a 2-approximation of the problem Weighted Vertex Cover.