MIT 6.7220/15.084 — Nonlinear Optimization (Spring '25) Tue, Feb 25th 2025

Lecture 6
Separation as a proof system

Instructor: Prof. Gabriele Farina (✉ gfarina@mit.edu)*

Beyond the ellipsoid method, separation also gives rise to deep results about *certification* of infeasibility for optimization problems. To appreciate those, however, we need to make a digression to appreciate the notion of *complexity class* for a problem.

L6.1 Decision problems and complexity

Within computer science, one of the main goals of *complexity theory* is that of classifying problems into *complexity classes.*

L6.1.1 Decision problems

For the purposes of this lecture, we can focus our attention on *decision problems*, that is, those problems for which we seek to construct an algorithm whose output is either `true` or `false`. Examples of decision problems include the following:

- `PRIME`: given as input an integer n (encoded in binary), return `true` if n is a prime number, and `false` otherwise.
- `FACTOR`: given as input two integers n and m (encoded in binary), return `true` if n has an integer divisor in the range $[2, m]$, and `false` otherwise.
- `LP`: given a set $Ax \leq b$ of m inequalities in n variables with rational coefficients (encoded as fractions of numbers encoded in binary), return `true` if the set $\{x \in \mathbb{R}^n : Ax \leq b\}$ is nonempty, and `false` otherwise.

Note that while these are decision problems, if we knew that a correct algorithm correctly solves the problem in time polynomial in the input, we could use such an algorithm to *solve* a search problem.

- For example, if `FACTOR` was known to be solvable in polynomial time, then we could factorize an integer m by binary searching the largest factor n, divide m by n, and repeat, taking a total of $\log^2(m)$ times the runtime of the algorithm.
- Similarly, when faced with a generic linear program

*These notes are class material that has not undergone formal peer review. The TAs and I are grateful for any reports of typos.

$$\begin{aligned} \min_x \quad & c^\top x \\ \text{s.t.} \quad & Ax \le b \\ & x \in \mathbb{R}^n, \end{aligned}$$

we could first establish a range $[a, b]$ such that, for sure, the optimal value would fall within the range. Then, we could find the value $c^\top x$ at optimality by performing a binary search on the interval $[a, b]$. At each iteration, we could check if at optimality $c^\top x \le \gamma$ by using the algorithm for `LP` to check whether the system of inequalities

$$Ax \le b, \quad c^\top x \le \gamma$$

has a solution. If yes, then we would decrease γ; if not, then we would have overshot on our guess, and we would need to increase γ.[1] Armed with the value of the optimum, we could then perform another binary search for each coordinate of x.

L6.1.2 Complexity classes for decision problems

Let's recall three fundamental complexity classes.

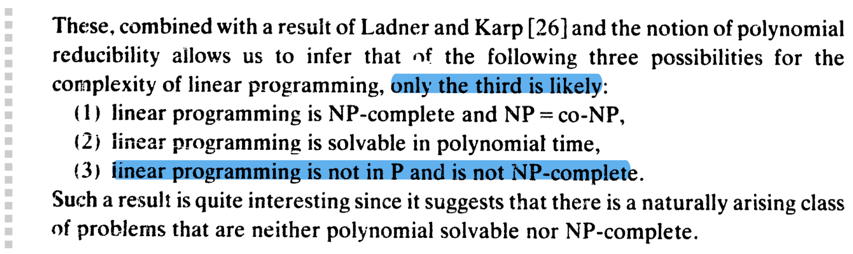
- A decision problem is in the complexity class P if there exists an algorithm that in polynomial time (in the size of the input description) returns the correct answer.
- A decision problem is in the complexity class NP if for all `true` instances there exists a polynomially-sized (in the size of the input description) certificate that can be given as input to a polynomial-time *verification algorithm*. The verification algorithm takes as input the problem instance and the certificate, and outputs `true` if and only if the certificate proves that the decision `true` on the problem instance is indeed correct.
- A decision problem is in the complexity class co-NP if for all `false` instances there exists a polynomially-sized (in the size of the input description) certificate that can be given as input to a polynomial-time *verification algorithm*. The verification algorithm takes as input the problem instance and the certificate, and outputs `true` if and only if the certificate proves that the decision `false` on the problem instance is correct.

Remark L6.1. It is clear from definition that every problem in P is automatically in NP and also in co-NP. So, $\mathsf{P} \subseteq \mathsf{NP} \cap \text{co-}\mathsf{NP}$. A *major* open question in complexity theory is proving whether $\mathsf{P} = \mathsf{NP} \cap \text{co-}\mathsf{NP}$. The general consensus is that $\mathsf{P} \neq \mathsf{NP} \cap \text{co-}\mathsf{NP}$.

In light of the remark, finding problems that are in $\mathsf{NP} \cap \text{co-}\mathsf{NP}$ and yet provably cannot be solved in polynomial time is a major open question. Linear programming is one example of problem famously in $\mathsf{NP} \cap \text{co-}\mathsf{NP}$ (we will see how such a result rests firmly on separation).

- For a long time, people conjectured that linear programming (`LP`) was *not* in P. See for example this excerpt from the introduction of a paper by Dobkin, D. P., & Reiss, S. P. [DR80]:

[1]Several details are missing from this description, but the main idea of using the binary search was the really important insight; the rest can be fixed. Can you think of how you could deal with an infeasible or unbounded linear program? How could one compute the interval $[a, b]$ in polynomial time in the input?



These, combined with a result of Ladner and Karp [26] and the notion of polynomial reducibility allows us to infer that of the following three possibilities for the complexity of linear programming, only the third is likely:

(1) linear programming is NP-complete and NP = co-NP,
(2) linear programming is solvable in polynomial time,
(3) linear programming is not in P and is not NP-complete.

Such a result is quite interesting since it suggests that there is a naturally arising class of problems that are neither polynomial solvable nor NP-complete.

They were proven wrong by the ellipsoid method. This should give a bit more context as to why the ellipsoid method was such a surprising development to warrant the front page of the New York Times (see Lecture 5).

- `PRIME` is another important problem that is known to be in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$ (this is not obvious, but with a bit of number theory it can be shown using only elementary results on cyclic groups[2]). `PRIME` was shown to also be in P in a breakthrough result by Agrawal, M., Kayal, N., & Saxena, N. [AKS04].
- Finally, also `FACTOR` is in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$. This problem is *not* currently known to be in P.

L6.2 Linear programming belongs to $\mathsf{NP} \cap \mathsf{co\text{-}NP}$

It is pretty straightforward that `LP` is in NP. This is because if a system of inequalities $Ax \le b$ has a solution, then the solution itself is the certificate, and one can verify that the certificate is correct by carrying out the matrix-vector product Ax and checking that indeed $Ax \le b$.[3]

It is significantly less obvious that `LP` is in $\mathsf{co\text{-}NP}$, that is, that whenever $Ax \le b$ does *not* have a solution, we can still certify that in polynomial time with a polynomially-sized certificate.

How would you certify that $Ax \le b$ has *no* solution? Here is a case in which a polynomially-sized certificate can be given. Suppose that there exist *nonnegative* multipliers $y_1, ..., y_m$ for the m inequalities defined by $Ax \le b$ with the following property:

- Multiply each inequality $a_j^\top x \le b_j$ by y_j;
- Then, sum all inequalities, obtaining a new inequality in which the left-hand side is identically 0, and the right-hand side is (strictly) negative.

Then, the original system of inequalities was clearly unsatisfiable. In this case, the vector $y = (y_1, ..., y_m)$ is a valid certificate of infeasibility. One (perhaps unexpected?) consequence of separation is that the above certificate *must always exist when $Ax \le b$* is infeasible. This result typically goes under the name of *Farkas lemma.*

Theorem L6.1 (Farkas lemma)**.** Let $Ax \le b$ be a system of inequalities where $A \in \mathbb{R}^{m\times n}$. Then, exactly one of the following options is true:

- either $Ax \le b$ has a solution; or
- there exists a vector $y \ge 0$ such that $A^\top y = 0$ and $b^\top y < 0$.

[2]The existence of polynomially-sized certificates of primality was shown by Pratt, V. R. [Pra75].

[3]Here, we are ignoring subtleties related to the encoding of numbers on machines. This is not a problem in our case: if $Ax \le b$ has a solution, then it must have a rational solution encodable using polynomially many bits [▷ Try to prove this].

Proof. As mentioned, the proof of this result relies on separation. In particular, consider the set

$$\Omega := \left\{Ax + s : x \in \mathbb{R}^n, s \in \mathbb{R}^m_{\geq 0}\right\} \subseteq \mathbb{R}^m,$$

The set Ω is a convex cone. Furthermore, if $b \in \Omega$, then this means that $b = Ax^* + s^*$ for some $x^* \in \mathbb{R}^n$ and $s^* \geq 0$; so, $Ax^* = b - s^* \leq b$, which shows that $Ax \leq b$ has a solution. On the other hand, if $b \notin \Omega$, then we can use separation!

In particular, if $b \notin \Omega$, we know that there must exist $u \in \mathbb{R}^m$ such that

$$\langle u, b\rangle < 0 \qquad \text{and} \qquad \langle u, Ax + s\rangle \geq 0 \quad \forall x \in \mathbb{R}^n, s \in \mathbb{R}^m_{\geq 0}.$$

Setting $s = 0$ but letting x be arbitrary in $\mathbb{R}^n$, we have

$$\langle u, Ax\rangle \geq 0 \quad \forall x \in \mathbb{R}^n \qquad \Longleftrightarrow \qquad \langle A^\top u, x\rangle \geq 0 \quad \forall x \in \mathbb{R}^n.$$

Since x is arbitrary, the only vector $A^\top u$ that can possibly satisfy such a condition is $A^\top u = 0$. Hence, the vector $u \in \mathbb{R}^n$ that arises from separation serves as a valid certificate y.

Finally, setting $x = 0$ and $s = e_i \geq 0$, where e_i is the i-th indicator vector,[4] we find that

$$\langle u, e_i\rangle \geq 0.$$

Since i was arbitrary, this shows that $u \geq 0$, completing the proof of existence of the certificate of infeasibility.

This shows that either the first bullet or the second bullet holds. To complete the proof, we need to show that it is not possible that they both hold. This is trivial: if $A^\top y = 0$ and $b^\top y < 0$, then no solution to $Ax \leq b$ can possibly exist, as that would imply that $0 = (y^\top A)x = y^\top(Ax) \leq y^\top b < 0$, a contradiction. □

Bibliography for this lecture

[DR80] Dobkin, D. P., & Reiss, S. P. (1980). The complexity of linear programming. *Theoret. Comput. Sci.*, *11*(1), 1–18. `https://doi.org/10.1016/0304-3975(80)90031-6`

[Pra75] Pratt, V. R. (1975). Every Prime Has a Succinct Certificate. *SIAM Journal on Computing*, *4*(3), 214–220. `https://doi.org/10.1137/0204018`

[AKS04] Agrawal, M., Kayal, N., & Saxena, N. (2004). PRIMES is in P. *Ann. Of Math.*, *160*(2), 781–793. `https://doi.org/10.4007/annals.2004.160.781`

Changelog

- `Feb 25, 2025: Simplified proof of Farkas lemma using conic separation.`

[4]That is, the vector containing all zeros except in position i, where it has a one.