Appendix F

A randomized Algorithm for Linear Programming

Let us recall the setup from last lecture: we have a linear program of the form

$$\begin{array}{lll} \text{(LP)} & \text{maximize} & c^\top x \\ & \text{subject to} & Ax \leqslant b, \end{array} \tag{F.1}$$

where $c, x \in \mathbb{R}^d$ (there are d variables), $b \in \mathbb{R}^n$ (there are n constraints), and $A \in \mathbb{R}^{n \times d}$. The scenario that we are interested in here is that d is a (small) constant, while n tends to infinity.

The goal of this lecture is to present a randomized algorithm (due to Seidel [2]) for solving a linear program whose expected runtime is $O(n)$. The constant behind the big-O will depend exponentially on d, meaning that this algorithm is practical only for small values of d.

To prepare the ground, let us first get rid of the unboundedness issue. We add to our linear program a set of $2d$ constraints

$$-M \leqslant x_i \leqslant M, \quad i = 1, 2, \ldots d, \tag{F.2}$$

where M is a symbolic constant assumed to be larger than any real number that it is compared with. Formally, this can be done by computing with rational functions in M (quotients of polynomials of degree d in the "variable "M), rather than real numbers. The original problem is bounded if and only if the solution of the new (and bounded) problem does not depend on M. This is called the *big-M method.*

Now let H, $|H| = n$, denote the set of original constraints. For $h \in H$, we write the corresponding constraint as $a_h x \leqslant b_h$.

Definition F.3. *For $Q, R \subseteq H$, $Q \cap R = \emptyset$, let $x^*(Q, R)$ denote the lexicographically largest optimal solution of the linear program*

$$\begin{array}{llll} \textit{LP(Q,R)} & \textit{maximize} & c^\top x & \\ & \textit{subject to} & a_h x \leqslant b_h, & h \in Q \\ & & a_h x = b_h, & h \in R \\ & & -M \leqslant x_i \leqslant M, & i = 1, 2, \ldots, d. \end{array}$$

If this linear program has no feasible solution, we set $x^*(Q, R) = \infty$.

What does this mean? We delete some of the original constraints (the ones not in $Q \cup R$, and we require some of the constraints to hold with equality (the ones in R). Since every linear equation $a_h x = b_h$ can be simulated by two linear inequalities $a_h x \leqslant b_h$ and $a_h x \geqslant b_h$, this again assumes the form of a linear program. By the big-M method, this linear program is bounded, but it may be infeasible. If it is feasible, there may be several optimal solutions, but choosing the lexicographically largest one leads to a unique solution $x^*(Q, R)$.

Our algorithm will compute $x^*(H, \emptyset)$, the lexicographically largest optimal solution of (F.1) subject to the additional bounding-box constraint (F.2). We also assume that $x^*(H, \emptyset) \neq \infty$, meaning that (F.1) is feasible. At the expense of solving an auxiliary problem with one extra variable, this may be assumed w.l.o.g. (Exercise).

Exercise F.4. *Suppose that you have an algorithm $\mathcal{A}$ for solving* feasible *linear programs of the form*

$$\begin{array}{lll} \text{(LP)} & \text{maximize} & c^\top x \\ & \text{subject to} & Ax \leqslant b, \end{array}$$

where feasible means that there exists $\tilde{x} \in \mathbb{R}^d$ such that $A\tilde{x} \leqslant b$. Extend algorithm $\mathcal{A}$ such that it can deal with arbitrary (not necessarily feasible) linear programs of the above form.

F.1 Helly's Theorem

A crucial ingredient of the algorithm's analysis is that the optimal solution $x^*(H, \emptyset)$ is already determined by a constant number (at most d) of constraints. More generally, the following holds.

Lemma F.5. *Let $Q, R \subseteq H, Q \cap R = \emptyset$, such that the constraints in R are* independent*. This means that the set $\{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$ has dimension $d - |R|$.*

If $x^*(Q, R) \neq \infty$, then there exists $S \subseteq Q, |S| \leqslant d - |R|$ such that

$$x^*(S, R) = x^*(Q, R).$$

The proof uses *Helly's Theorem*, a classic result in convexity theory.

Theorem F.6 (Helly's Theorem [1]). *Let $C_1, \ldots C_n$ be $n \geqslant d + 1$ convex subsets of $\mathbb{R}^d$. If any $d + 1$ of the sets have a nonempty common intersection, then the common intersection of* all *n sets is nonempty.*

Even in $\mathbb{R}^1$, this is not entirely obvious. There it says that for every set of intervals with pairwise nonempty overlap there is one point contained in all the intervals. We will not prove Helly's Theorem here but just use it to prove Lemma F.5.

Proof. (**Lemma** F.5) The statement is trivial for $|Q| \leqslant d - |R|$, so assume $|Q| > d - |R|$. Let

$$L(R) := \{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$$

and

$$B := \{x \in \mathbb{R}^d : -M \leqslant x_i \leqslant M, i = 1, \dots, d\}.$$

For a vector $x = (x_1, \dots, x_d)$, we define $x_0 = c^\top x$, and we write $x > x'$ if $(x_0, x_1, \dots, x_d)$ is lexicographically larger than $(x'_0, x'_1, \dots, x'_d)$.

Let $x^* = x^*(Q, R)$ and consider the $|Q| + 1$ sets

$$C_h = \{x \in \mathbb{R}^d : a_h x \leqslant b_h\} \cap B \cap L(R), \quad h \in Q$$

and

$$C_0 = \{x \in \mathbb{R}^d : x > x^*\} \cap B \cap L(R).$$

The first observation (that may require a little thinking in case of C_0) is that all these sets are convex. The second observation is that their common intersection is empty. Indeed, any point in the common intersection would be a feasible solution $\tilde{x}$ of $LP(Q, R)$ with $\tilde{x} > x^* = x^*(Q, R)$, a contradiction to $x^*(Q, R)$ being the lexicographically largest optimal solution of $LP(Q, R)$. The third observation is that since $L(R)$ has dimension $d - |R|$, we can after an affine transformation assume that all our $|Q| + 1$ convex sets are actually convex subsets of $\mathbb{R}^{d-|R|}$.

Then, applying Helly's Theorem yields a subset of $d - |R| + 1$ constraints with an empty common intersection. Since all the C_h do have $x^*(Q, R)$ in common, this set of constraints must contain C_0. This means, there is $S \subseteq Q, |S| = d - |R|$ such that

$$x \in C_h \ \forall h \in S \quad \Rightarrow \quad x \notin C_0.$$

In particular, $x^*(S, R) \in C_h$ for all $h \in S$, and so it follows that $x^*(S, R) \leqslant x^* = x^*(Q, R)$. But since $S \subseteq Q$, we also have $x^*(S, R) \geqslant x^*(Q, R)$, and $x^*(S, R) = x^*(Q, R)$ follows. □

F.2 Convexity, once more

We need a second property of linear programs on top of Lemma F.5; it is also a consequence of convexity of the constraints, but a much simpler one.

Lemma F.7. *Let* $Q, R \subseteq H, Q \cap R \neq \emptyset$ *and* $x^*(Q, R) \neq \infty$. *Let* $h \in Q$. *If*

$$a_h x^*(Q \setminus \{h\}, R) > b_h,$$

then

$$x^*(Q, R) = x^*(Q \setminus \{h\}, R \cup \{h\}).$$

Before we prove this, let us get an intuition. The vector $x^*(Q \setminus \{h\}, R)$ is the optimal solution of $LP(Q \setminus \{h\}, R)$. And the inequality $a_h x^*(Q \setminus \{h\}, R) > b_h$ means that the constraint h is violated by this solution. The implication of the lemma is that at the optimal solution of $LP(Q, R)$, the constraint h must be satisfied with equality in which case this optimal solution is at the same time the optimal solution of the more restricted problem $LP(Q \setminus \{h\}, R \cup \{h\})$.

Proof. Let us suppose for a contradition that

$$a_h x^*(Q, R) < b_h$$

and consider the line segment s that connects $x^*(Q, R)$ with $x^*(Q \setminus \{h\}, R)$. By the previous strict inequality, we can make a small step (starting from $x^*(Q, R)$) along this line segment without violating the constraint h (Figure F.1). And since both $x^*(Q, R)$ as well as $x^*(Q \setminus \{h\}, R)$ satisfy all other constraints in $(Q \setminus \{h\}, R)$, convexity of the constraints implies that this small step takes us to a feasible solution of $LP(Q, R)$ again. But this solution is lexicographically larger than $x^*(Q, R)$, since we move towards the lexicographically larger vector $x^*(Q \setminus \{h\}, R)$; this is a contradiction. □

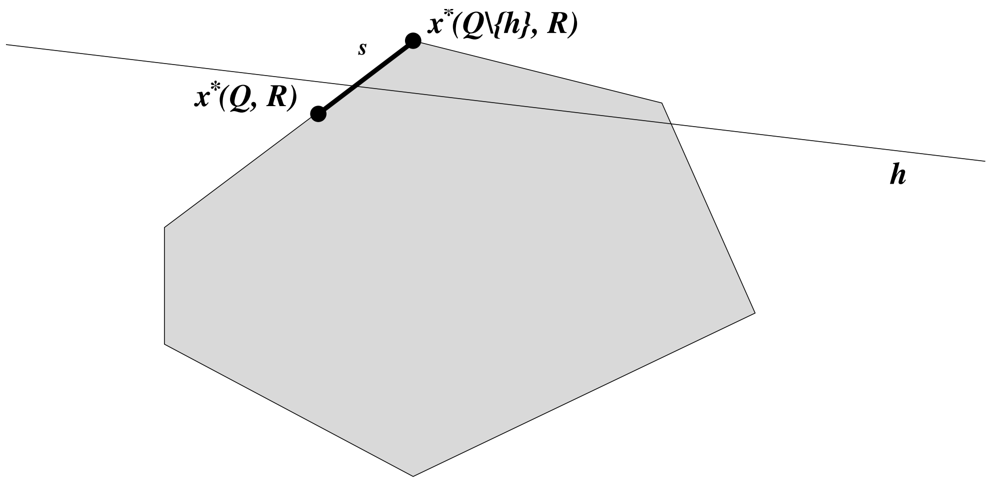

Figure F.1: *Proof of Lemma F.7*

F.3 The Algorithm

The algorithm reduces the computation of $x^*(H, \emptyset)$ to the computation of $x^*(Q, R)$ for various sets Q, R, where R is an *independent* set of constraints. Suppose you want to compute $x^*(Q, R)$ (assuming that $x^*(Q, R) \neq \infty$). If $Q = \emptyset$, this is "easy", since we have a constant-size problem, defined by R with $|R| \leqslant d$ and $2d$ bounding-box constraints $-M \leqslant x_i \leqslant M$.

Otherwise, we choose $h \in Q$ and recursively compute $x^*(Q \setminus \{h\}, R) \neq \infty$. We then check whether constraint h is violated by this solution. If not, we are done, since then $x^*(Q \setminus \{h\}, R) = x^*(Q, R)$ (Think about why!). But if h is violated, we can use Lemma F.7 to conclude that $x^*(Q, R) = x^*(Q \setminus \{h\}, R \cup \{h\})$, and we recursively compute the latter solution. Here is the complete pseudocode.

```
$\mathcal{LP}(Q, R)$:
    IF $Q = \emptyset$ THEN
        RETURN $x^*(\emptyset, R)$
    ELSE
        choose $h \in Q$ uniformly at random
        $x^* := \mathcal{LP}(Q \setminus \{h\}, R)$
        IF $a_h x^* \leqslant b_h$ THEN
            RETURN $x^*$
        ELSE
            RETURN $\mathcal{LP}(Q \setminus \{h\}, R \cup \{h\})$
        END
    END
```

To solve the original problem, we call this algorithm with $\mathcal{LP}(H, \emptyset)$. It is clear that the algorithm terminates since the first argument Q becomes smaller in every recursive call. It is also true (Exercise) that every set R that comes up during this algorithm is indeed an independent set of constraints and in particular has at most d elements. The correctness of the algorithm then follows from Lemma F.7.

Exercise F.8. *Prove that all sets R of constraints that arise during a call to algorithm $\mathcal{LP}(H, \emptyset)$ are independent, meaning that the set*

$$\{x \in \mathbb{R}^d : a_h x = b_h, h \in R\}$$

of points that satisfy all constraints in R with equality has dimension $d - |R|$.

F.4 Runtime Analysis

Now we get to the analysis of algorithm LP, and this will also reveal why the random choice is important.

We will analyze the algorithm in terms of the expected number of *violation tests* $a_h x^* \leqslant b_h$, and in terms of the expected number of *basis computations* $x^*(\emptyset, R)$ that it performs. This is a good measure, since these are the dominating operations of the algorithm. Moreover, both violation test and basis computation are “cheap” operations in the sense that they can be performed in time $f(d)$ for some f.

More specifically, a violation test can be performed in time $O(d)$; the time needed for a basis computation is less clear, since it amounts to solving a small linear program itself. Let us suppose that it is done by brute-force enumeration of all vertices of the bounded polyhedron defined by the at most $3d$ (in)equalities

$$a_h x = b_h, h \in R$$

and

$$-M \leqslant x_i \leqslant M, i = 1, \ldots, d.$$

F.4.1 Violation Tests

Lemma F.9. *Let* $T(n, j)$ *be the maximum expected number of violation tests performed in a call to* $\mathcal{LP}(Q, R)$ *with* $|Q| = n$ *and* $|R| = d - j$. *For all* $j = 0, \ldots, d$,

$$\begin{aligned} T(0, j) &= 0 \\ T(n, j) &\leqslant T(n-1, j) + 1 + \frac{j}{n} T(n-1, j-1), \quad n > 0. \end{aligned}$$

Note that in case of $j = 0$, we may get a negative argument on the right-hand side, but due to the factor $0/n$, this does not matter.

Proof. If $|Q| = \emptyset$, there is no violation test. Otherwise, we recursively call $\mathcal{LP}(Q \setminus \{h\}, R)$ for some h which requires at most $T(n-1, j)$ violation tests in expectation. Then there is one violation test within the call to $\mathcal{LP}(Q, R)$ itself, and depending on the outcome, we perform a second recursive call $\mathcal{LP}(Q \setminus \{h\}, R \cup \{h\})$ which requires an expected number of at most $T(n-1, j-1)$ violation tests. The crucial fact is that the probability of performing a second recursive call is at most j/n.

To see this, fix some $S \subseteq Q, |S| \leqslant d - |R| = j$ such that $x^*(Q, R) = x^*(S, R)$. Such a set S exists by Lemma F.5. This means, we can remove from Q all constraints except the ones in S, without changing the solution.

If $h \notin S$, we thus have

$$x^*(Q, R) = x^*(Q \setminus \{h\}, R),$$

meaning that we have already found $x^*(Q, R)$ after the first recursive call; in particular, we will then have $a_h x^* \leqslant b_h$, and there is no second recursive call. Only if $h \in S$ (and this happens with probability $|S|/n \leqslant j/n$), there can be a second recursive call. □

The following can easily be verified by induction.

Theorem F.10.

$$T(n, j) \leqslant \sum_{i=0}^{j} \frac{1}{i!} j! n.$$

Since $\sum_{i=0}^{j} \frac{1}{i!} \leqslant \sum_{i=0}^{\infty} \frac{1}{i!} = e$, we have $T(n, j) = O(j! n)$. If $d \geqslant j$ is constant, this is $O(n)$.

F.4.2 Basis Computations

To count the basis computations, we proceed as in Lemma F.9, except that the "1" now moves to a different place.

Lemma F.11. *Let $B(n, j)$ be the maximum expected number of basis computations performed in a call to $\mathcal{LP}(Q, R)$ with $|Q| = n$ and $|R| = d - j$. For all $j = 0, \ldots, d$,*

$$\begin{aligned} B(0, j) &= 1 \\ B(n, j) &\leqslant B(n-1, j) + \frac{j}{n} B(n-1, j-1), \quad n > 0. \end{aligned}$$

Interestingly, $B(n, j)$ turns out to be much smaller than $T(n, j)$ which is good since a basic computation is much more expensive than a violation test. Here is the bound that we get.

Theorem F.12.

$$B(n, j) \leqslant (1 + H_n)^j = O(\log^j n),$$

where H_n is the n-th Harmonic number.

Proof. This is also a simple induction, but let's do this one since it is not entirely obvious. The statement holds for $n = 0$ with the convention that $H_0 = 0$. It also holds for $j = 0$, since Lemma F.11 implies $B(n, 0) = 1$ for all n. For $n, j > 0$, we inductively obtain

$$\begin{aligned} B(n, j) &\leqslant (1 + H_{n-1})^j + \frac{j}{n}(1 + H_{n-1})^{j-1} \\ &\leqslant \sum_{k=0}^{j} \binom{j}{k} (1 + H_{n-1})^{j-k} \left(\frac{1}{n}\right)^k \\ &= (1 + H_{n-1} + \frac{1}{n})^j = (1 + H_n)^j. \end{aligned}$$

The second inequality uses the fact that the terms $(1 + H_{n-1})^j$ and $\frac{j}{n}(1 + H_{n-1})^{j-1}$ are the first two terms of the sum in the second line. □

F.4.3 The Overall Bound

Putting together Theorems F.10 and F.12 (for $j = d$, corresponding to the case $R = \emptyset$), we obtain the following

Theorem F.13. *A linear program with n constraints in d variables (d a constant) can be solved in time $O(n)$.*

Questions

85. *What is Helly's Theorem?* Give a precise statement and outline the application of the theorem for linear programming (Lemma F.5).

86. *Outline an algorithm for solving linear programs*! Sketch the main steps of the algorithm and the correctness proof! Also explain how one may achieve the preconditions of feasibility and boundedness.

87. *Sketch the analysis of the algorithm!* Explain on an intuitive level how randomization helps, and how the recurrence relations for the expected number of violation tests and basis computations are derived. What is the expected runtime of the algorithm?

References

[1] Herbert Edelsbrunner, *Algorithms in combinatorial geometry*, vol. 10 of *EATCS Monographs on Theoretical Computer Science*, Springer, 1987.

[2] Raimund Seidel, Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, **6**, (1991), 423–434.