CMPUT 675: Topics in Combinatorics and Optimization **Fall 2016**

Lecture 24 (Nov 2): Solving LPs via Separation Oracles

Lecturer: Zachary Friggstad *Scribe: Huijuan Wang*

24.1 Feasibility

We discussed how the Ellipsoid method can be used in a much more general setting to potentially solve linear programs with exponentially many variables. The main thing we need is to be able to answer the question of whether a given point is in $\mathcal{P}$ or not and finding a separating hyperplane in the latter case. A procedure which does this is called separation oracle. Next we will focus on **Separation Oracles**.

24.2 Separation Oracles

Definition 1 *Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a convex set. A separation oracle for $\mathcal{P}$ is an algorithm which when given $x \in \mathbb{Q}^n$ outputs one of the following:*
① **Feasibile***: Just a declaration that $x \in \mathcal{P}$ (if $x \in \mathcal{P}$).*
② **Separating hyperplane***: Return some $a \in \mathbb{R}^n$ such that $a^{\mathrm{T}} z > a^{\mathrm{T}} x$, for all $z \in \mathcal{P}$ (if $x \notin \mathcal{P}$).*
The running time should be polynomial in n and the bit complexity of x.

We will discuss how to solve a linear program in polynomial time when we are only given the objective function and a separation oracle as input. Low-level details of the proofs will be skipped, we focus more on the core ideas. Details can be found in the course textbook [2] and further information in [1].

24.2.1 Outline

We begin by solving a slightly relaxed goal.

Definition 2 (Weak Optimization Problem) *Given $\mathcal{P} \in \mathbb{R}^n$ convex, find $x \in \mathcal{P}$, such that $c^{\mathrm{T}} x \geqslant \sup_{z \in \mathcal{P}} c^{\mathrm{T}} z - \epsilon$.*

Of course, we want a true optimum extreme point. First, we want such a statement to coincide with actually being near some extreme point. This is not true if there are multiple extreme points, but we can slightly perturb the objective function to achieve this.

Assumption
There is a unique extreme point optimum to $\max\{c^T \cdot x : x \in \mathcal{P}\}$.

This can assumed without loss of generality perturbing c by a small amount: by adding $(\delta, \delta^2, \delta^3, \ldots, \delta^n)^T$ to c where $\delta > 0$ is a tiny value in $\mathbb{Q}$ whose bit complexity depends polynomially on n and the largest possible entries of A, b and c (but not the number of constraints). The straightforward details are in [2], but the idea is that there is enough space between different extreme point values such that a suboptimal extreme point remains

suboptimal while the slight tilting of c ensures no facet of $\mathcal{P}$ is optimal on all points unless it is a corner already: so there is a unique optimum.

Following this, we will have that solving the weak optimization problem coincides with actually being near the optimum extreme point. Then we will describe a number-theoretic routine to snap the near optimum solution to the nearest fraction with denominators at most Γ (recall this was our symbol for our Hadamard bound on the bit complexity of extreme points). This will snap our near optimum solution to a true optimum extreme point solution.

24.2.2 The Grötschel - Lovász - Schrijver Algorithm

We make one further assumption that is not without loss of generality. It can be lifted, but it requires even more work. See [1] for details.

Our assumption is that the set of feasible solutions contains a ball of some positive radius (i.e. it is "full dimensional"). Furthermore, we assume we already know such a ball (in particular, we know a feasible solution).

The algorithm simply does the following. It runs the Ellipsoid method, but if it finds a feasible solution, the hyperplane used to guide the next step is simply the objective function vector itself. This way, the next ellipsoid is enclosing all points with value at least the value of this feasible guess. It returns the best solution found over all iterations.

We also suppose we know all feasible solutions are contained in a ball of radius $R > 0$. The algorithm iterates until the volume of the ellipsoid shrinks to an appropriately small value ϵ'.

Algorithm 1 Algorithm

$M \leftarrow R^2 \cdot \mathbf{I}$
$x \leftarrow x^*$ {Our known initial solution}
Invariant: $E(M, x)$ contains all $\tilde{x}$ such that $c^{\mathrm{T}}\tilde{x} \geqslant c^{\mathrm{T}}x^*$.
while vol $E(M, x) > \epsilon^{'}$ **do**
  if $x \in \mathcal{P}$ **then**
    $a \leftarrow c$
    if $c^{\mathrm{T}}x \geqslant c^{\mathrm{T}}x^*$ **then**
      $x^* \leftarrow x$
    end if
  else
    $a \leftarrow$ separating hyperplane from the oracle
    $Mx \leftarrow$ Löwner-John Ellipsoid for M, x, a;
  end if
end while
return x^*

We only provide the intuition on why the returned value x^* is a near-optimum solution. Consider Figure 24.1. The shaded region is all solutions with value close to the optimum (within $-\epsilon$ of the optimum). If the polytope is full-dimensional, then its volume has polynomial bit complexity.

The invariant ensures the ellipsoid contains all solutions with value $\geq c^T x^*$. If $c^T x^* < OPT - \epsilon$ then the volume of the ellipsoid is at least the volume of the shaded region. So for an appropriate choice of ϵ' (which can be explicitly calculated in a simple ways to depend only on the maximum bit complexity of coefficients in A, b, c and n, see [2] for details) we must have that x^* is within the shaded region.

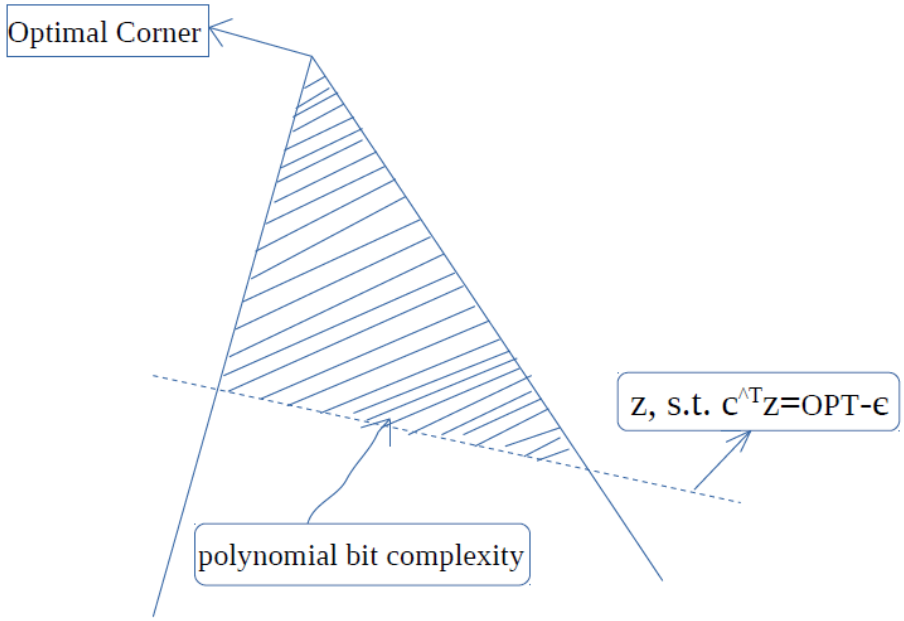

Figure 24.1: Proof

24.3 Snapping to An Extreme Point

Recall, again, that $\Gamma \in \mathbb{Z}$ denotes our upper bound on the numerator and denominator of an extreme point of $\mathcal{P}$.

We begin by outlining our objective. The proof is in [2], the idea is that the difference between distinct fractions with denominators at most Γ is at least $\frac{1}{\Gamma^2}$.

Lemma 1 *Let x^* be a solution to the weak optimization problem for some appropriate choice of $\epsilon > 0$ (with polynomial bit complexity). Let $\overline{x}$ be the optimal extreme point (we are assuming it is unique). For all $1 \leq i \leq n$, the nearest fraction $\frac{p}{q}$ to x_i^*, such that $q \leqslant \Gamma$ is $\overline{x_i}$.*

Proof.[**sketch**] For distinct $a, b \in \mathbb{Q}$ having denominators at most Γ we have $|a-b| \geq \frac{1}{\Gamma^2}$ (by cross multiplying).

For small enough $\epsilon > 0$, having x^* in the shaded region of Figure 24.1 means $\sum_{i=1}^{n} |\overline{x}_i - x_i^*| < \frac{1}{2\Gamma^2}$ (there are a few low-level details to verify here). In particular, it should hold on a term-by-term basis so the nearest fraction to x_i^* with denominator at most Γ is then $\overline{x_i}$.

■

So, we are left solving the following problem to finally snap our near-optimal solution x^* to a true optimum.

Definition 3 (Nearest-Fraction Problem) *Given $p \geq 0$, $q \geq 1$, $\Gamma \geq 1$, find the fraction $\frac{a}{b}$ with $b \leq \Gamma$ nearest to $\frac{p}{q}$.*

We solve this with a gcd-like algorithm. It is reminiscent of the Extended Euclidean Algorithm for expressing the gcd of two numbers as an integer linear combination of those numbers. But the calculations are slightly different. Really, this is more closely related to computing the continued fraction expansion of p/q.

We compute a sequence of values q_i, r_i, s_i, t_i for $i \geq 0$ (we ignore q_i for $i = 0$). Initialize them as follows.

i	r_i	s_i	t_i
0	p	0	1
1	q	1	0

Then for $i \geq 1$ and as long as $r_i \neq 0$, we compute the following:

- **Quotient**: $q_i = \lfloor r_{i-1}/r_i \rfloor$
- **Remainder**: $r_{i+1} = r_{i-1} - q_i \cdot r_i$
- **Numerator**: $s_{i+1} = s_{i-1} + q_i \cdot s_i$
- **Denominator**: $t_{i+1} = t_{i-1} + q_i \cdot t_i$

Think that the sequence s_i/t_i provides better and better approximations of p/q. We will show this formally, and stopping at some carefully chosen i and using an approximation like this will give us the answer.

24.3.1 Invariants

① $s_{i+1} \cdot t_i - s_i \cdot t_{i+1} = (-1)^i$, for $i \geq 0$.

This is easy to prove by induction. It holds initially (examine the table). Inductively,

$$s_{i+1} \cdot t_{i+2} - s_{i+2} \cdot t_{i+1} = s_{i+1} \cdot (t_i + q_{i+1} \cdot t_{i+1}) - (s_i + q_{i+1} \cdot s_{i+1}) \cdot t_{i+1} = s_{i+1} \cdot t_i - s_i \cdot t_{i+1} + 0 = (-1)^i.$$

② $\dfrac{r_i \cdot s_i + r_{i+1} \cdot s_{i+1}}{r_i \cdot t_i + r_{i+1} \cdot t_{i+1}} = \dfrac{p}{q}$ for $i \geq 0$.

③ $\dfrac{s_i}{t_i} \geq \dfrac{p}{q}$ for odd $i \geq 0$.

④ $\dfrac{s_i}{t_i} \leq \dfrac{p}{q}$ for even $i \geq 0$.

The remaining three invariants are also proven by induction in a fairly straightforward manner (if we regard $1/0$, one of the base cases, as being greater than p/q).

24.3.2 Running Time

The sequence of remainders just follows the remainders from Euclid's algorithm, so the algorithm terminates after $O(\log q)$ steps. The s and t sequences are increasing. Consider the final index i we computed, so $r_i = 0$. Invariant ② then shows $s_{i-1}/t_{i+1} = p/q$ so they stay bounded in bit complexity throughout. So the algorithm runs in polynomial time (in the bit complexity of the inputs).

24.3.3 Constructing the Best Fraction

If $q \leq \Gamma$ already then we don't have to do anything. Otherwise, let i be the largest index such that $t_i \leq \Gamma$. This is well-defined as the previous paragraph argues $s_{i^*-1}/t_{i^*-1} = p/q$ where i^* is such that $r_{i^*} = 0$ and we are assuming $q > \Gamma$. Let α be the maximum value that $t_{i-1} + \alpha t_i \leq \Gamma$, note by our choice of i it must be $\alpha < r_i$.

Let $y = \frac{s_i}{t_i}$ and $z = \frac{s_{i-1}+\alpha s_i}{t_{i-1}+\alpha t_i}$. Return whichever of the two is closer to p/q, by our choice of i and α both values have denominator at most Γ.

It is fairly simple to see p/q lies between y and z: for example if i is even the invariant shows $y \le p/q \le \frac{s_{i+1}}{t_{i+1}} \le z$ (the last bound is straightforward to verify). We show every fraction strictly between y and z has denominator $> \Gamma$, so one of y, z is the nearest fraction to p/q with denominator at most Γ we are looking for.

To that end, let a/b be a fraction strictly between y and z. On one hand, invariant ① shows

$$|y - z| = \left|\frac{\pm 1}{t_i(t_{i-1} + \alpha t_i)}\right| = \frac{1}{t_i(t_{i-1} + \alpha t_i)}.$$

On the other hand, because $p/q \neq y, z$ we then have

$$|y - z| = |y - a/b| + |a/b - z| \ge \frac{1}{qt_i} + \frac{1}{b(t_{i-1} + \alpha t_i)} = \frac{t_{i-1} + (\alpha + 1)t_i}{bt_i(t_{i-1} + \alpha t_i)}.$$

Comparing both expressions above and rearranging, $b \ge t_{i-1} + (\alpha + 1)t_i > \Gamma$ by our choice of α.

References

M. Grötschel, L. Lovasz, and A. Schrijver, Geometric Algorithms and Combinatorial Optimization, *Springer-Verlag Berlin Heidelberg*, 1993.

B. Korte and J. Vygen, Combinatorial Algorithms - Theory and Algorithms, *Springer-Verlag, Berlin Heidelberg*, 2012.