15-451: Algorithms October 22, 2019

Lecture Notes: Computational Geometry: 2D-LP

Lecturer: Gary Miller *Scribes:*

[1]

1 Introduction

1.1 Definitions

Definition 1.1. (Linear Programming) Linear programming (LP) are problems that can be expressed in canonical form as

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{d} \end{aligned}$$

where $A \in \mathbb{R}^{n\times m}$, $\mathbf{x} \in \mathbb{R}^{m\times 1}$, $\mathbf{c} \in \mathbb{R}^{m\times 1}$, and $\mathbf{d} \in \mathbb{R}^{n\times 1}$.

Note that $\mathbf{x} \leq \mathbf{y}$ if $\forall i, x_i \leq y_i$.

Definition 1.2. (Feasible) The LP region is feasible if $\exists \mathbf{x}, A\mathbf{x} \leq \mathbf{d}$.

Note that $\{\mathbf{x}|A\mathbf{x} \leq \mathbf{d}\}$ the feasible region is convex.

1.2 2D LP Example

In 2D case, LP is expressed as follows:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}, \ \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

[1] Originally 15-750 notes by Yanzhe Yang and Yao Liu

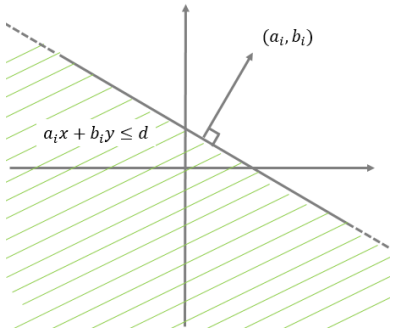

Figure 1: h_i is half-plane normal to (a_i, b_i)

2D LP can be interpreted from geometry view. Firstly, we define Half-plane or Half-space.

$$h_i \equiv \{(x, y) | a_i x + b_i y \leq d_i\}$$

is Half-plane or Half-space. As shown in Figure 1, h_i is half-plane normal to (a_i, b_i).
From geometric view, the input to 2D LP are:

- Half-planes $\{h_1, h_2, ..., h_n\}$
- Vector $\mathbf{c} \in \mathbb{R}^2$

The goal of LP is to find farthest x_i in c direction. Here $x \in \bigcap_{i=1}^{n} h_i$.

For simplification, we will exclude some corner cases in following discussion.

1. No h_i is normal to $\mathbf{c}$
2. $\bigcap_{i=1}^{n} h_i$ is a bounded feasible region
3. "Bounding" box m_1, m_2 will be given (shown in figure 2). Note that m_1, m_2 are half-spaces.

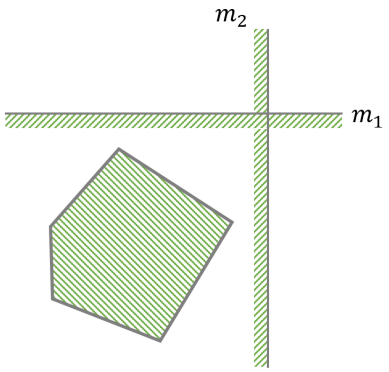

Figure 2: A bounding box is defined by half space m_1 and m_2.

1.3 1D LP

1D LP can be expressed as

$$\begin{array}{ll} \max & cx \\ \text{subject to} & a_i x \leq b_i, a_i \neq 0 \end{array}$$

Note that the inputs of 1D LP are constraints $a_i x \leq b_i, a_i \neq 0$. Without loss of generality, $a_i = \pm 1$.

Constraints can be grouped as 2 types, C^+ and C^- (shown in Figure 3).

$$\begin{aligned} C^+ &= \{i | x \leq b_i\} \\ C^- &= \{i | -x \leq b_i\}, \text{i.e. } -b_i \leq x \end{aligned}$$

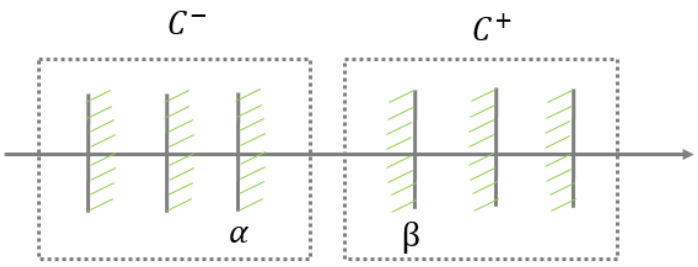

Figure 3: h_i is half-plane normal to (a_i, b_i)

Denote $\alpha = \max\{-b_i | i \in C^-\}$, $\beta = \min\{b_i | i \in C^+\}$.

From Figure 3, we can see a LP region is feasible if and only if $\alpha \leq \beta$. If it is feasible, return

$$f(n) = \begin{cases} c\ \beta & \text{if sign}(c) = 1 \\ c\ \alpha & \text{otherwise} \end{cases}$$

Theorem 1.3. *1D LP is $O(n)$ time.*

1.4 Cost

If we look at these computational geometry algorithms, there are two types: one is linear runtime like selection, and the other is $O(n \log n)$ runtime like sorting, convex hull, half-space intersection, and meshing. What we discussed today, LP in fixed dimensions, lies on the linear runtime side.

We expect LP in fixed dimensions to be linear as it is like selection, which asks for the k largest elements. Details on runtime analysis is shown in Section 2.2.

$O(n \log n)$	$O(n)$
Sorting Convex Hull Half-space Intersection Meshing	Selection LP(fixed dim)

Table 1: Algorithm Costs

2 Random Incremental 2D Algorithm

In this section, we will use the 1D linear programming algorithm as an oracle to construct the 2D linear programming algorithm.

Algorithm 1 2D random incremental LP algorithm

Input: $m_1, m_2, h_1, h_2, \ldots, h_n, c$
Output: 2D-LP(h_1, h_2, $\ldots$, h_n, c)

Step 1: $v_0 \leftarrow$ 2D-LP(m_1, m_2, c) (i.e. $v_0 = CH(m_1) \cap CH(m_2)$)
Step 2: Randomly order h_1, h_2, $\ldots$, h_n
Step 3:
for $i = 1$ to n **do**
    if $v_{i-1} \in h_i$ **then**
        $v_i \leftarrow v_{i-1}$
    else(Make and solve 1D-LP problem)
        L $\leftarrow CH(h_i)$ (Boundary of h_i)
        for $j = 1$ to $i - 1$ **do**
            $h_j^{'} \leftarrow L \cap h_j$
        end for
        $c^{'} \leftarrow$ projection(c, L) [2](Note: $c^{'} \neq 0$)
        $v_i \leftarrow$ 1D-LP($h_1^{'}$, $h_2^{'}$, $\ldots$, $h_{i-1}^{'}, c^{'}$)
    end if
    if v_i is "undefined" **then**
        Report "No Solution" and halt
    end if
end for
return v_n

2.1 Proof of correctness

Claim 2.1. *At any time, v_i = LP($m_1, m_2, h_1, h_2, \ldots, h_i, c$)*

Proof. We will do induction on i. The base case is trivial, if we are given the oracle 2D-LP(m_1, m_2, c), the optimum is the intersection of the boundaries of m_1 and m_2.

Assume v_{i-1} is correct. There are two cases for v_i:

Case 1: $v_{i-1} \in h_i$. Then v_{i-1} is in the feasible region of the new problem. Thus v_{i-1} is the optimal solution to the problem.

Case 2: $v_{i-1} \notin h_i$. We claim that the solution to the new problem, v_i, must be on the boundary of h_i, $CH(h_i)$. If not, assume there is an optimal solution opt_i to the problem LP($m_1, m_2, h_1, h_2, \ldots, h_i, c$), which is not on $CH(h_i)$. Since $v_{i-1} \notin h_i$ and $opt_i \in h_i$, then the line segment connecting v_{i-1} and opt_i must intersect $CH(h_i)$ at some point p.
Since v_{i-1} is the optimal solution without constraint of h_i, we have that

$$c^T v_{i-1} \geq c^T opt_i$$

[2]Here projection is a function mapping a point in 2D space into a line.

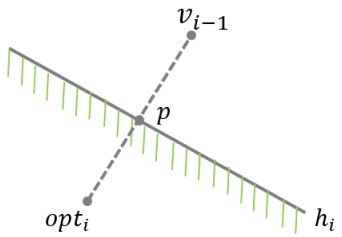

Figure 4: The line of v_{i-1} and opt_i must intersect with $CH(h_i)$ at some point p

Then according to the monotonic property of linear function, we have that

$$c^T p \geq c^T opt_i$$

which means $p \in CH(h_i)$ is also an optimal solution. Thus there must be an optimal solution in $CH(h_i)$. Then we solve the linear programming constrained on $CH(h_i)$ to get v_i. Thus v_i has the optimal value and is feasible. □

2.2 Runtime analysis

Claim 2.2. *2D-LP is $O(n)$ expected time.*

Proof. We will prove that claim by backward analysis. Suppose we remove h_j from $h_1, h_2, \ldots, h_i$.

Definition 2.3. (Critical point) h_j is critical if and only if removing it changes optimal solution.

When the optimal solution is an intersection of two boundaries, then the corresponding two constraints are critical. If the optimal solution is an intersection of 3 boundaries, then we can verify that there is at most 2 critical points.

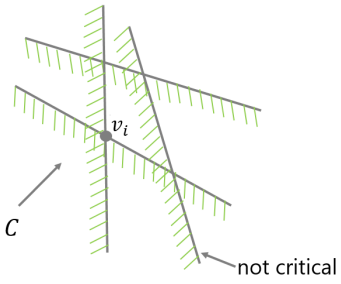

Figure 5: 2 critical constraints

So, there are at most 2 critical constraints. When h_j is not critical, we have that $v_{i-1} \in h_i$ and the cost is a constant k. When h_j is critical, then the algorithm will solve a 1D LP problem of i constraints and the cost is $i \cdot k$. Thus the worst case for Step 3 is exactly 2 critical constraints.

Let E_i be the expected cost of step 3 at time i:

$$E_i \leq \left(\left(\frac{2}{i} \right) ki + \left(\frac{i-2}{i} \right) k \right) \leq 3k$$

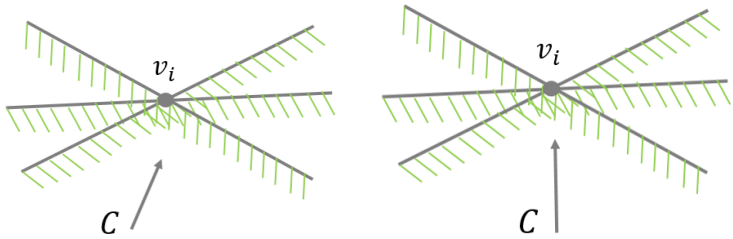

Figure 6: Left: 1 critical constraint; Right: 0 critical constraint

Thus the total expected work is:

$$\sum_{i=1}^{n} 3k = O(n)$$

□

2.3 Determining Unboundness or Finding the Bounding Box

Lemma 2.4. *A LP problem is unbounded if and only if:*

1. *It is feasible*
2. $\exists$ *d s.t.* $c^T\mathbf{d} > 0$ *and* $A\mathbf{d} \leq 0$

Proof.

($\Longleftarrow$) By the first condition, there exists $\bar{\mathbf{x}}$ such that $A\bar{\mathbf{x}} \leq b$. Pick any $\alpha \geq 0$ and $\mathbf{d}$ satisfying the second condition. Now:

$$A(\bar{\mathbf{x}} + \alpha\mathbf{d}) \leq \alpha A\mathbf{d} + A\bar{\mathbf{x}} \leq \alpha A\mathbf{d} + b \leq b$$

Thus $\bar{\mathbf{x}}+\alpha\mathbf{d}$ is feasible for any $\alpha > 0$. The objective $c^T(\bar{\mathbf{x}}+\alpha\mathbf{d}) = \alpha c^T\mathbf{d}+c^T\bar{\mathbf{x}}$ goes to infinity with α.

($\Longrightarrow$) This can be proved by compactness. □

2.3.1 Finding d for unboundedness

Now we are going to derive a method to determine unboundedness and feasibility. When LP is bounded, we will output the bounding box. When it is unbounded, we need to find out $\mathbf{d}$.

We know that $\mathbf{d}$ exist if and only if $\exists$ $\mathbf{d}$ s.t. $\mathbf{c}^T\mathbf{d} = 1$ and $A\mathbf{d} \leq 0$ (figure 7). Note that all the constraint boundaries cross the origin. By projecting all the constraints to the line $c^T\mathbf{d} = 1$, it is a 1D-LP problem.

Note that there exist $\mathbf{d}$ if and only if the 1D-LP has a feasible solution, which means $\alpha > \beta$. Here, $\alpha = \max\{-b_i | i \in C^-\}$, $\beta = \min\{b_i | i \in C^+\}$. (Figure 3)

Claim 2.5. *If* $\beta > \alpha$ *and* $A\mathbf{x} \leq \mathbf{b}$ *is feasible, then the LP is unbounded and we can find a* $\mathbf{d}$.

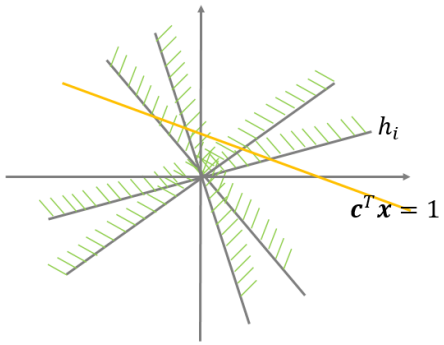

Figure 7: $\mathbf{c}^T\mathbf{d} = 1$, $A\mathbf{d} \leq 0$

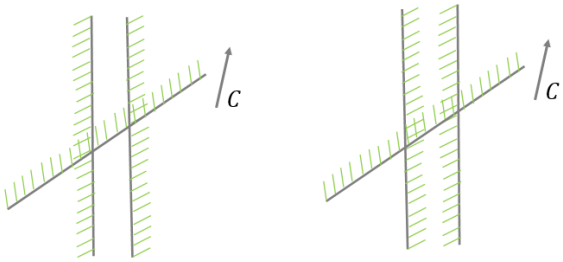

Figure 8: $\beta = \alpha$

Claim 2.6. *If $\beta = \alpha$ (shown in figure 8), there are two parallel constraints and $A\mathbf{x} \leq \mathbf{b}$ may be feasible or infeasible. We need to recover $A\mathbf{x} \leq \mathbf{b}$ to determine the feasibility. When the LP is feasible it is unbounded and there is exactly one $\mathbf{d}$.*

Claim 2.7. *If $\beta < \alpha$, let h_α be the half-plane giving α, and h_β be the half-plane giving β, then (h_α, h_β) is one bounding box.*