A Generic Reference: Ryan recommends a very nice book “Understanding and Using Linear Programming” [MG06].

1 Introduction

Linear Programming (LP) refers to the following problem. We are given an input of the following $m$ constraints (inequalities):

$$K \subseteq \mathbb{R}^n = \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m \end{cases}$$

where every $a_{ij}, b_i \in \mathbb{Q}$, for $i \in [m], j \in [n]$. Our goal is to determine whether a solution exists, i.e., $K \neq \emptyset$, or to maximize $c_1x_1 + \cdots + c_nx_n$ for some $c_1 \cdots c_n \in \mathbb{Q}$ (we work toward the first goal for now; we’ll talk about the second goal later). If $K \neq \emptyset$ we output a point in $\mathbb{Q}^n$ in $K$; if $K = \emptyset$ we output a “proof” that it’s empty. We’ll prove that if $K \neq \emptyset$ we can always find a rational solution $x^*$ (i.e., $x^* \in \mathbb{Q}^n$) in $K$, and define what a proof of unsatisfiability is in the next section. A remark of this problem:

Remark 1.1. We can change the “$\geq$” in the definition to “$\leq$” since $a \cdot x \leq b \iff -a \cdot x \geq -b$. We can also allow equality since $a \cdot x = b \iff a \cdot x \geq b \land a \cdot x \leq b$. However, we CANNOT allow “$>$” or “$<$”.

The significance of this problem is:

Theorem 1.2. [Kha79] LP is solvable in polynomial-time.

2 Fourier-Motzkin Elimination

If some (non-negative) linear combinations of the $m$ input constraints give us $0 \geq 1$, then clearly the LP is unsatisfiable (we emphasize non-negative so that we don’t change the direction of inequality). Thus we define:
Definition 2.1. A proof of unsatisfiability is $m$ multipliers $\lambda_1 \cdots \lambda_m \geq 0$ such that the sum over $i \in [m]$ of $\lambda_i$ times the $i^{th}$ constraint gives us $0 \geq 1$, i.e.,

$$\begin{align*}
\lambda_1 a_{1i} + \lambda_2 a_{2i} + \cdots + \lambda_m a_{mi} &= n \\
\lambda_1 b_1 + \lambda_2 b_2 + \cdots + \lambda_m b_m &= 1
\end{align*}$$

Here’s a few remarks to this definition:

Remark 2.2. We don’t need $0 \geq 1$; $0 \geq q$ for any positive $q$ would suffice. However we may divide every $\lambda_i$ by $q$ to get $0 \geq 1$.

Remark 2.3. If our LP algorithm outputs a solution for $K \neq \emptyset$ but outputs nothing for $K = \emptyset$, we can use this algorithm to generate the $\lambda_i$’s, since we can see from our definition that the existence of these $\lambda_i$’s is an LP (with $m$ variables and $n + 1$ constraints).

It turns out that if an LP is unsatisfiable, a proof exists.

Theorem 2.4 (Farkas Lemma/LP duality). $K \neq \emptyset \implies$ such $\lambda_i$’s exist.

We’ll “prove” the result with an example of Fourier-Motzkin Elimination.

Proof. With loss of generality (WLOG), assume we are working with a 3-variable 5-constraint LP, namely,

$$\begin{align*}
x - 5y + 2z &\geq 7 \\
3x - 2y - 6z &\geq -12 \\
-2x + 5y - 4z &\geq -10 \\
-3x + 6y - 3z &\geq -9 \\
-10y + z &\geq -15
\end{align*}$$

Our first step is to eliminate $x$. We do this by multiplying each constraint with a positive constant such that the coefficient of $x$ in each constraint is either -1, 1, or 0 (if an original constraint has 0 as the coefficient of $x$ we just leave it alone). In our example, we get:

$$\begin{align*}
x - 5y + 2z &\geq 7 \\
x - \frac{2}{3} y - 2z &\geq -4 \\
-x + \frac{5}{2} y - 2z &\geq -5 \\
-x + 2y - z &\geq -3 \\
-10y + z &\geq -15
\end{align*}$$

We then write the inequalities as $x \geq c_iy + d_i z + e_i$ or $x \leq c_iy + d_i z + e_i$ for some $c_i, d_i, e_i \in \mathbb{Q}$, depending on whether $x$ has coefficient 1 or -1 (again, if $x$ has coefficient 0 we just leave it alone) in the inequality, i.e.,

$$\begin{align*}
x &\geq 5y - 2z + 7 \\
x &\geq \frac{2}{3} y + 2z - 4 \\
x &\leq \frac{5}{2} y - 2z + 5 \\
x &\leq 2y - z + 3 \\
-10y + z &\geq -15
\end{align*}$$
In order to satisfy the inequality, for each pair \( x \geq c_i y + d_i z + e_i, x \leq c_j y + d_j z + e_j \), we must have \( c_j y + d_j z + e_j \geq c_i y + d_i z + e_i \). Without changing the satisfiability of the original constraints, we change them (again, we leave alone those without \( x \)) into the new ones, i.e.,

\[
\begin{align*}
\frac{5}{2}y - 2z + 5 & \geq 5y - 2z + 7 \\
\frac{5}{2}y - 2z + 5 & \geq \frac{2}{3}y + 2z - 4 \\
2y - z + 3 & \geq 5y - 2z + 7 \\
2y - z + 3 & \geq \frac{2}{3}y + 2z - 4 \\
-10y + z & \geq -15
\end{align*}
\]

In this way we eliminate \( x \). We can repeat this process until we have only 1 variable left. In general we’ll end up with inequalities of the following form:

\[
\begin{align*}
x_n & \geq -3 \\
x_n & \geq -6 \\
\cdots \\
x_n & \leq 10 \\
x_n & \leq -1 \\
\cdots \\
0 & \geq -2 \\
0 & \geq -10 \\
\cdots 
\end{align*}
\]

Then the original LP is satisfiable if and only if the maximum of all \( q_i \) in inequalities \( x_n \geq q_i \) is less than the maximum of all \( q_j \) in inequalities \( x_n \leq q_j \), and every \( q_k \) in the inequalities \( 0 \geq q_k \) is non-positive. All inequalities derived are non-negative linear combinations of original constraints, so every \( q_i, q_j, q_k \) above are rational numbers.

Assume for now that the original constraints are satisfiable. Then we’ll be able to pick a \( x_n \in \mathbb{Q} \) that satisfies the inequalities we end up with. Substituting the \( x_n \) to the inequalities we get one step ago, we’ll get some (satisfiable) inequalities of \( x_{n-1} \), so we can keep back-substituting until we get a solution \( x^* = (x_1, x_2, \cdots, x_n) \in \mathbb{Q}^n \). This proves that if an LP has solution it must also have a rational one.

We now assume that the original constraints are not solvable, so if we do another step of Fourier-Motzkin Elimination, we’ll end up with \( 0 \geq c \) for some positive \( c \), which is a non-negative linear combination of original constraints, thus proving the result.

Notice that Fourier-Motzkin Elimination actually solves LP; however, it’s not polynomial. During each step, if we start with \( k \) inequalities, in the worst case we may end up with \( k^2/4 = \Theta(k^2) \) new inequalities. Since we start with \( m \) constraints and must perform \( n \) steps, in the worst case the algorithm may take \( \Theta(m^{3n}) \) time, which is far too slow. We’ll show how LP can be solved efficiently in the next section.
3 Equational form

Our next goal will be to solve LP in polynomial time. However, it is not immediately clear that LP is solvable in polynomial time; for example, if every solution of an LP requires exponentially-many bits to write down, then it would be impossible to solve LP in poly-time. In this section we’ll prove that this will not be the case, i.e., if an LP is satisfiable, then we can always find a solution with size polynomial to input size.

Theorem 3.1. Given input

\[
K = \begin{cases} 
    a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\
    \cdots \\
    a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m 
\end{cases}
\]

with input size (number of bits needed to write the input) \(L\), which we denote as \(\langle K \rangle = L\),

(1) \(K \neq \emptyset \implies \exists \text{ feasible solution } x^* \in \mathbb{Q}^n \text{ with } \langle x^* \rangle = \text{poly}(L);\)

(2) \(K = \emptyset \implies \exists \text{ proof } \lambda_i's \text{ with } \langle \lambda_i \rangle = \text{poly}(L) \text{ for every } i \in [m].\)

Proof. Suffices to prove (1) (by Remark 2.3). Assume \(K\) is not empty. If we consider the geometric meanings of our constraints, we may see that the set of solutions is a closed convex set in \(\mathbb{R}^n\). Here is an example in \(\mathbb{R}^2\):

![Figure 1: A Geometric view of LP](image)

If \(K \neq \emptyset\), then there exists some feasible vertices/extreme points/basic feasible solutions, which are the unique intersections of some linearly independent EQUATIONS out of the \(m\) EQUATIONS (we emphasize “equations” because constraints are originally given as inequalities, but we are now considering their equational form. Any vertex \(x^*\) of our solution set is a solution to a \(n \times n\) system of equations, which can be found (and written down, of course) in polynomial time by Gaussian Elimination, and this proves that \(\langle x^* \rangle = \text{poly}(L)\). \(\square\)
This proof basically sketches what we are trying to do, but it has problems; consider the following $n = 2, m = 1$ case:

![Figure 2: $x_1 + x_2 \geq 1$](image)

We don’t have any vertices in it! Then how can we find a polynomial-size solution? We observe, however, that we can solve the example above by adding the 2 axes. Before we move on, we give a definition:

**Definition 3.2.** We say $K$ is included by a big box if $\forall i \in [n], x_i \leq B_i^+, x_i \geq B_i^-, \langle B_i^+ \rangle, \langle B_i^- \rangle = \text{poly}(L)$.

**Observation 3.3.** The proof above would be fine if $K$ is included by a big box.

We now give another proof:

**Proof.** For each constraint $a^{(i)} x \geq b_i$, we replace it with $a^{(i)} x - S_i = b_i$ and $S_i \geq 0$, where the $S_i$’s are called slack variables. Then, replace each original $x_i$ with $x_i^+ - x_i^-$ and $x_i^+ \geq 0, x_i^- \geq 0$. Then the new LP has $2n + m$ variables, each constrained to be non-negative, and all the other constraints are equations. We call this converted “Equational Form” of LP $K'$. Then $K'$ has exactly the same solutions with $K$ regarding the original variables, and $\langle K' \rangle = \text{poly}(\langle K \rangle)$. We write $K'$ as $K' = \begin{cases} A'x' = b' \\ x' \geq 0 \end{cases}$, where $A'$ is a $m' \times n'$ matrix, where $n' = 2n + m$, and $m' = m$. Assume WLOG that $A'$ has rank $m'$ (otherwise some constraints are unnecessary, so we can throw them out). Then $K'$ is contained in a positive orthant of $\mathbb{R}^n$, and $A'x' = b'$ is an $(n' - m')$-dimensional subspace in $\mathbb{R}^n$. If $m' = n'$ then this subspace is a point, and this point $x^*$ is a feasible solution with size $\text{poly}(L)$, so we’re done. Otherwise $K$ is the intersection of this subspace and the positive orthant of $\mathbb{R}^n$. Since the subspace will not be parallel to all coordinates, it must intersect with some axes. Let the intersection be $x^*$. Then $x^*$ is a solution to $K'$ and $\langle x^* \rangle = \text{poly}(L)$.  

\[ \blacksquare \]
To sum up what we have shown, given an LP problem that asks whether \( K \) is empty, we can, WLOG, include into \( K \) a big box, where each \( \langle B_i^- \rangle, \langle B_i^+ \rangle = \text{poly}(L) \), and if \( K \neq \emptyset \), then we can always find a vertex \( x^* \) in \( K \).

4 LP and reduction

Assume we already have a polynomial-time program that decides LP, i.e., it outputs “Yes” if \( K \neq \emptyset \) and outputs “No” if \( K = \emptyset \).

Q: How can we use this program as a black box to solve LP (i.e., outputs a point in \( Q^n \) in \( K \) if \( K \neq \emptyset \)?)

A: Suppose \( K \neq \emptyset \). We only need to find \( a_1, a_2, \ldots a_n \in Q \) such that \( K \cap \{ \forall i \in [n] x_i = a_i \} \neq \emptyset \). How? From the last section we know that we can assume every variable \( x_i \) is bounded by \( B_i^- \) and \( B_i^+ \), where \( \langle B_i^- \rangle = \text{poly}(L), \langle B_i^+ \rangle = \text{poly}(L) \). Then for each \( x_i \), we can do a binary search, starting with \( B_i = (B_i^- + B_i^+)/2 \). If our program tells us \( K \cap \{ B_i \leq x_i \leq B_i^+ \} = \emptyset \), we know that \( B_i^- \leq x \leq B_i^+ \); otherwise we search whether \( (B_i^- + B_i^+)/2 \leq x \leq B_i^- \), etc. The binary search takes at most \( O(\log_2 B_i^+ - B_i^-) = O(\log_2 2^{\text{poly}(L)}) = \text{poly}(L) \) time for each variable, so we can efficiently find a point in \( K \).

Q: What if \( K \) is something like \( \{ 3x_1 = 1 \} \)? In that case the binary search may never end.

A: WLOG assume every \( a_{ij} \in N \) for \( i \in [m], j \in [n] \) and \( b_i \in N \) for \( i \in [n] \). Let \( c = \prod |a_{ij}| \). Since each \( a_{ij} \) has size \( L \), i.e., \( |a_{ij}| \leq 2^L \), we have \( c \leq 2^{mnL} = 2^{\text{poly}(L)} \). Then if \( K \neq \emptyset \), there must be some vertex \( v \) such that \( cv_i \in Z \) for every \( i \in [n] \), and \( cv_i \) is bounded by \( cB_i^- \) and \( cB_i^+ \), both of which still has size \( \text{poly}(L) \). Then we can do binary search efficiently.

Recall that another goal of LP is to maximize \( c \cdot x \) for some \( c \in Q^n \) with the constraints \( K \).

Q: Given a program that decides LP, How do we solve \( \text{max}\{ c \cdot x : x \in K \} \)?

A: We can add \( \{ c \cdot x \geq \beta \} \) as a constraint and do binary search to determine the largest \( \beta \) such that \( K \cap \{ c \cdot x \geq \beta \} \neq \emptyset \}. \) We have to be a bit careful here, however, because the maximum can be infinity. To fix this, we have the following observation.

Observation 4.1. Suppose \( c \) is not parallel to any \( a_i \). If \( \text{max}\{ c \cdot x \} \) is not infinity, then the maximum must occur at a vertex.

All vertices are bounded by \( B_i^- \) and \( B_i^+ \), so we can bound \( \text{max}\{ c \cdot x \} \) with some \( M \) by choosing \( B_i = B_i^+ \) if \( c_i > 0 \) and \( B_i = B_i^- \) if \( c_i < 0 \) and set \( M = c \cdot B \). Now that the maximum is bounded, we can compute \( \text{max}\{ c_i x_i | x \in K \cup \{ c \cdot x \leq M + 1 \} \} \) by the way we described above. If the maximum we get is \( M + 1 \), then it is actually infinity; otherwise we get the maximum we intended.
Now that we know how to maximize $c \cdot x$, we switch back to the first goal, and suppose $K \neq \emptyset$.

Q: Instead of any $x \in K$, how can we make sure we output a vertex?
A: From the observation, we may choose an arbitrary $c \in \mathbb{Q}^n$ that is not parallel to any constraints, and maximize $c \cdot x$. This guarantees to give us a vertex.

References
