Linear Programming

1 Intro to Linear Programming

1.1 The Linear Programming Problem

LINEAR PROGRAMMING (LP): Given a matrix $A \in \mathbb{R}^{n \times m}$ and a vector $\vec{b} \in \mathbb{R}^n$, find $\vec{x} \in \mathbb{R}^m$ with $A \cdot \vec{x} \leq \vec{b}$ (keep in mind that the $\leq$ operation for vectors is coordinate-wise). Visually:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m \leq b_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m \leq b_n$$

While we define LP as having $\leq$ constraints, notice that it allows for $\geq$ and $=$ constraints as well.

$$\vec{a} \cdot \vec{x} \leq b \iff (-\vec{a}) \cdot \vec{x} \geq -b$$

$$\vec{a} \cdot \vec{x} \leq b, \vec{a} \cdot \vec{x} \geq b \iff \vec{a} \cdot \vec{x} = b$$

Thus, our LP algorithm will be at least as powerful as Gaussian elimination, which can solve systems of just equality constraints.

1.2 Canonical Forms

We can introduce an optimization constraint (maximizing or minimizing some linear function $\vec{c} \cdot \vec{x}$) which doesn't make the problem very much harder, and is much more useful to us. The two canonical LP forms are:

(1). Maximize $\vec{c} \cdot \vec{x}$ subject to $A \cdot \vec{x} \leq \vec{b}$ with $\vec{x} \geq 0$.

(2). Minimize $\vec{c} \cdot \vec{x}$ subject to $A \cdot \vec{x} \geq \vec{b}$ with $\vec{x} \geq 0$.

1.3 Computational Complexity

Theorem 1. *If there is a solution $\vec{x}$ to an LP, then there is one with a number of bits polynomial in m, n, and the number of bits in $\vec{c}, \vec{b}$,and A. (Difficult proof studied in CS 675).*

Theorem 2. *LP* $\in$ ***NP***.

Certificate: A proposed $\vec{x}$.

Certifier: An algorithm which ensures $A \cdot \vec{x}$ satisfies the inequality and $\vec{x} \geq 0$.

Notice that it is possible for $\vec{x}$ to have exponentially many bits (and thus our certificate would not be polynomially sized), but by Theorem 1, there will always be an $\vec{x}$ with polynomially many bits.

Theorem 3. *LP* $\in$ ***co-NP***.

Proof by Duality (covered later this class).

Theorem 4. *LP* $\in$ ***P***.

This is a difficult theorem due to Khachiyan and Karmarker in the 1980s; the proof is covered in CS 675.

1.4 The Diet Problem

The diet problem is the canonical example for LPs. Let there be three important nutrients denoted protein, carbs, and vitamins with requirements r_p, r_c, r_v. Furthermore, we have three key foods denoted chocolate, nachos, and ramen with prices $\vec{p} =< p_c, p_n, p_r >$. We have a matrix $U_{c/n/r,p/c/v}$ denoting the units of nutrients per unit of food, and our goal is to find the minimum cost way to satisfy our dietary needs. Let $\vec{x} =< x_c, x_n, x_r >$ be the amount of each food we buy, then we have an LP:

Minimize $\vec{p} \cdot \vec{x}$ subject to $U \cdot \vec{x} \geq \vec{r}$ with $\vec{x} \geq 0$.

1.5 Oracles and You (or, how to solve an LP while blindfolded)

It is possible to solve an LP with exponentially or even infinitely many constraints in polynomial time. That is, we can solve an LP *without looking at all the constraints.* Sound too good to be true? Well, it is: but not if we have oracles.

An **oracle** is a polynomial time function which gives us specific information about our LP, which we can then use to solve an LP in polynomial time. Khachiyan and Karmarkar discovered we need two oracles for this to be possible:

(1). Membership oracle: Given a vector $\vec{x}$, decides in polynomial time if $\vec{x}$ violates any constraint.

(2). Separation oracle: Given a vector $\vec{x}$ that violates at least one constraint, returns in polynomial time an arbitrary violated constraint.

Theorem 5. *If we have a membership oracle and a separation oracle, we can solve any LP in polynomial time.*

2 Max-Flow LPs

2.1 Max-Flow LP

Max-flow can be cleverly represented as a linear program. If we have a flow f_e on each edge e, then our LP is:

Maximize

$$\sum_{e \text{ out of } s} f_e$$

subject to

$$\begin{cases} f_e \leq c_e & \forall e \\ \sum_{e \text{ into } v} f_e = \sum_{e \text{ out of } v} f_e & \forall v \neq s, t \\ f_e \geq 0 & \forall e \end{cases}$$

2.2 Multi-Commodity Flow

MULTI-COMMODITY FLOW: Given source-sink pairs (s_i, t_i) with demands d_i, edge capacities c_e, is it possible to send d_i units of flow from each s_i to its corresponding t_i without violating any capacity constraints with the joint flows?

Linear programming is, as far as anyone knows, the only way to solve multi-commodity flow. Let $f_{e,i}$ represent the amount of flow of commodity i on each edge e. Then, the LP is:

Satisfy the constraints:

$$\begin{cases} \sum_{e \text{ into } t_i} f_{e,i} \geq d_i & \forall i \\ \sum_{e \text{ into } v} f_{e,i} = \sum_{e \text{ out of } v} f_{e,i} & \forall i, v \neq s_i, t_i \\ \sum_i f_{e,i} \leq c_e & \forall e, i \\ f_{e,i} \geq 0 & \forall e, i \end{cases}$$

3 Dualism

3.1 Lower Bounds on LP Solutions

Suppose we want to minimize $\vec{c} \cdot \vec{x}$ subject to $A\vec{x} \geq \vec{b}$ with $\vec{x} \geq 0$.

For any $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, if $\sum \alpha_i a_{ij} \leq c_j \ \forall j$, then $\sum \alpha_i \cdot b_i$ is a lower bound on the optimal solution of the LP. To get the best lower bound, we maximize $\sum_i a_i \cdot b_i$ subject to $\sum \alpha_i \cdot a_{ij} \leq c_j \ \forall j$ and $\alpha_i \geq 0 \ \forall i$. In other words, we have the LP:

Maximize $\vec{b} \cdot \vec{\alpha}$ subject to $A^T \cdot \vec{\alpha} \leq 0$ and $\vec{\alpha} \geq 0$.

3.2 Dualism Theorem

Theorem 6. *Let M be the LP: maximize $\vec{c} \cdot \vec{x}$ subject to $A\vec{x} \geq \vec{b}$ with $\vec{x} \geq 0$, and N be the LP: minimize $\vec{b} \cdot \vec{y}$ subject to $A^T \cdot \vec{y} \leq 0$ and $\vec{y} \geq 0$. Then, M and N are called **Duals** of each other.*

We have (1) Weak Duality and (2) Strong Duality.

(1). If $\vec{x}, \vec{y}$ are <u>any</u> solutions of M and N, then $\vec{c} \cdot \vec{x} \leq \vec{b} \cdot \vec{y}$.

(2). If $\vec{x}, \vec{y}$ are <u>optimal</u> solutions of M and N, then $\vec{c} \cdot \vec{x} = \vec{b} \cdot \vec{y}$.

Proof. Of (1); (2) is much more difficult and is covered in CS 675.

$$\vec{c} \cdot \vec{x} \leq (A^T \vec{y}) \cdot \vec{x} \text{ because } \vec{c} \leq A^T \text{ and } \vec{x} \geq 0$$

$$= \vec{y}^T (A \cdot \vec{x})$$

$$\leq \vec{y}^T \vec{b} \text{ because } A\vec{x} \leq \vec{b} \text{ and } \vec{y} \geq 0$$

□

3.3 Path Decomposition Linear Program

Since max-flow is an LP, it follows that the path decomposition form is also an LP. If we have a flow f_p on each path p, then our LP is:

Maximize

$$\sum_p f_p$$

subject to

$$\begin{cases} \sum_{p \ni e} f_p \leq c_e & \forall e \\ f_p \geq 0 & \forall p \end{cases}$$

Logically (yet elegantly) the Dual of the max-flow LP is the min-cut LP. Let x_e be the inclusion variable for each edge e. Then, the min-cut LP is:

Minimize

$$\sum_e c_e x_e$$

subject to

$$\begin{cases} \sum_{e \in p} x_e \geq 1 & \forall p \\ x_e & \forall e \end{cases}$$

Notice that this LP may have exponential size, but it is polynomial time solvable if we use Dijkstra's Algorithm as an oracle; it will check if the shortest path satisfies the first constraint in polynomial time.