Week 8 Recitation

1 Linear Programs

A linear program (LP) in the simplest term is asking for the simultaneous feasibility of a set of linear inequalities.

A linear inequality for some vector $x \in \mathbb{R}^n$ is a condition of the form of

$$\sum_{1 \le j \le n} a_j x_j \le b$$

A linear program is a collection of m of these, arranged as a matrix. So each of these conditions become

$$\sum_{1 \le j \le n} A_{i,j} x_j \le b_i$$

or in matrix notation, $Ax \le b$. Note that here $\le$ is overloaded to denote a vector being at most another entry-wise. The dimensions match here because A is $m \times n$, x is $n \times 1$, so b is $m \times 1$.

This 'list of inequalities' view encode other features of linear programs:

1. greater or equal: $-a^\top x \le -b$ (for $a, x \in \mathbb{R}^n$, $b \in \mathbb{R}$) is equivalent to $a^\top x \ge b$.
2. equalities: $a^\top x = b$ (for $a, x \in \mathbb{R}^n$, $b \in \mathbb{R}$) can be encoded as $a^\top x \le b$, $a^\top x \ge b$.
3. objective values: binary search, to maximize $c^\top x$ (for some cost vector c), check whether there is a solution with $c^\top x \ge \theta$ and binary search on the value of θ.

2 Cuts and Flows Recap

The (single commodity) flow linear program in the most succinct form is

$$\min_{\substack{f \in \mathbb{R}^E \\ B^\top f = d \\ 0 \le f_e \le \text{cap}_e \, \forall e \in E}} \quad c^\top f.$$

Note that the roles of m and n is flipped when we do flows,

Unraveled:

1. V is the set of vertices.
2. E is the set of directed edges, of the form $u \to v$, where u and v are vertices.

3. $f \in \mathbb{R}^E$, the flow, is a length E vector indicating the amount flowing on the edges.
4. The directions of the edges manifest in $0 \le f$.
5. The capacities manifest in $f \le \text{cap}$.
6. $d \in \mathbb{R}^V$ is the demand vector, indicating how much flow we want in/out of each vertex.
7. B is the edge vertex incidence matrix, $B^\top f$ gives the amount entering minus the amount leaving at each vertex, so $B^\top f = d$ enforces the amount in - amount out at all vertices meeting the demands.

A circulation is a flow f where $B^\top f = 0$.

Any circulation can be decomposed to a sum of cycles. The way Richard likes to think about this is to pair up all the in and out flows at each vertex locally, then argue that this is a collection of permutation.

Problem 1: Max $s \to t$ flow as MinCost Circulation

Express the maximum $s \to t$ flow problem as an instance of the mincost circulation problem (with capacities) mentioned in class. That is, given a graph, capacities cap, and source/sink vertices s/t, output a (possibly different) graph with capacities and costs c such that the maximum flow amount is exactly

$$\min_{\substack{f \in \mathbb{R}^E \\ 0 \le f \le \text{cap} \\ B^\top f = 0}} c^\top f.$$

3 Practice Problem

First verify that the minimum cut problem can also be written as a linear program.

Problem 2: Minimum Cut as a Linear Program

Show that in a directed graph with capacity cap, the fractional $s-t$ minimum cut linear program can written as

$$\min_{\substack{x \in \mathbb{R}^V, x_s = 0, x_t = 1 \\ y \in \mathbb{R}^E, y \ge 0 \\ y \ge Bx}} \text{cap}^\top y$$

where recall $B \in \mathbb{R}^{m \times n}$ is the edge-vertex incidence matrix, with an edge $e = (u \to v)$ corresponding to a row with $B_{e,u} = -1$, $B_{e,v} = 1$, and 0 everywhere else.

Solution. The x vector is the indicator for the vertices. $x_u = 0$ if it's on the same side as s, $x_u = 1$ if it's on the same side as t.

Then for an edge $u \to v$, $\max\{x_v - x_u, 0\}$ is whether the edge is cut. This value is given by minimizing

$$y_e \geq x_v - x_u$$
$$y_e \geq 0$$

so multiplying it by the capacity gives the capacity of this edge that's cut, fractionally. Summing this then gives the objective.

Problem 3: Traffic Routing Over Time

Consider a road network happening over t units of time. Each road now has two parameters, the time it takes to traverse it (which is an integer), and the cost of taking it. Furthermore, traffic congestion means that we can have at most 1 car passing through each vertex at any point of time. Given a demand vector d, we want to know whether it's possible to route it within the time limit, and subject to that, minimize the total cost of this routing. Formulate this as a minimum cost flow instance on a graph of size $\text{POLY}(n, t)$.

Solution. Split each vertex into two, and make a copy per time. That is, for each vertex u, create vertices (u, i, in) and (u, i, out) for each time $0 \leq i \leq t$.

Having an edge of capacity 1 and cost 0 from (u, i, in) to (u, i, out) for all times i ensures that at most one car passes through a vertex at any game.

Then, for an edge $u \to v$ with cost $c_{u\to v}$ and time duration $t_{u\to v}$, for all times i, add edges from (u, i, out) to $(v, i + t_{u\to v}, \text{in})$ with cost $c_{u\to v}$ and infinite capacity.

The minimum-cost flow in this network with demand vector d corresponds to the minimum cost of sending everyone under the time constraints. If the flow is not feasible, then there is also no way of sending everyone.