Intro to LP's

Thursday, November 05, 2015
4:38 AM

- So far, we've seen several big hammers to get algorithms:
  Divide & Conquer, Pruning, Dynamic Programming, Greedy
- Today, another big hammer: Linear Programming (LP)
- LP's solve optimization problems: maximize/minimize an objective subject to some constraints
- In LP's, objective is a linear function
  constraints are linear inequalities
- General form:

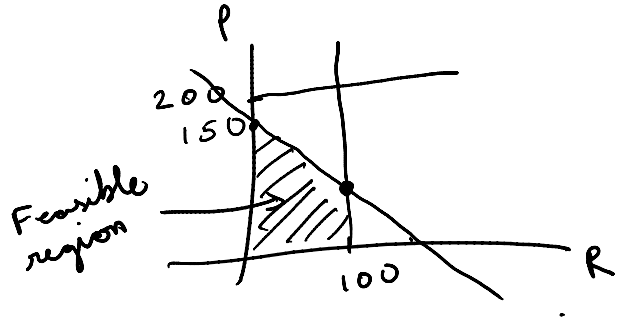
$$\max \; c^T x \qquad c \in \mathbb{R}^n$$
$$(*) \quad \text{s.t.} \; Ax \leq b \qquad A \in \mathbb{R}^{m \times n}, \; b \in \mathbb{R}^m$$
$$x \geq 0$$

- Example: Prakruti makes two kinds of idli, (P)lain and (R)awa. Plain idlis cost ₹1 each, Rawa idlis ₹2 each. Atmost 200 plains and 100 rawas can be made, and total number of idlis can be at most 150. How many plains and how many rawas should they make to max profit?

$$\max \; P + 2R$$
$$\text{s.t.} \; P \leq 200$$
$$R \leq 100$$
$$P + R \leq 150$$
$$P, R \geq 0$$

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} P \\ R \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} P \\ R \end{bmatrix} \leq \begin{bmatrix} 200 \\ 100 \\ 150 \end{bmatrix}$$



Optimum must be at a vertex!

Two types of degeneracies:

- Infeasible: Feasible region is empty.
  Ex: $x \geq 0$, $x \leq -1$.
- Unbounded: Arbitrary high objective value possible.

$$\max x \quad \text{s.t. } x \geq 0.$$

- LP's used to formulate many, many optimization problems!
- LP's can always be assumed to have above (*) form. Explain.
- Can also change all non-negativity inequalities to equalities. Exercise.

Solving LP's

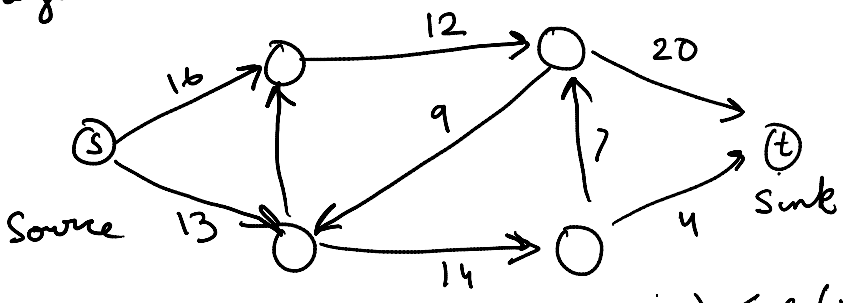
- In practice, LP's can be solved very fast by the simplex algorithm.
- Basic idea: start at a vertex, then move to a nbr'ing vertex such that objective increases. Keep going unless no such nbr.
- Vertex: a point where n inequalities are tight
- Nbr: Two vertices are nbr's if $n-1$ tight vertices are in common
- Note: can assume by perturbation that that every vertex is intersection of n hyperplanes.

$x_i > 0$. Then can increase x_i until one

inequality becomes tight (ordering vertices)
- Then restart process by transforming coordinates so that new vertex takes place of origin.
- Details in Dasgupta et al. Also see the case when origin is not feasible. (Section 7.6)

Maximum Flows

- Imagine transportation network with capacities on roads.



For all edges (u,v), $0 \le f(u,v) \le c(u,v)$

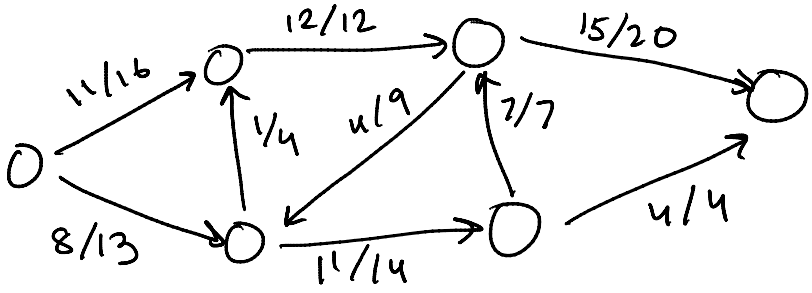
For all $u \in V \setminus \{s,t\}$: $\sum_{v:(v,u)\in E} f(v,u) = \sum_{v:(u,v)\in E} f(u,v)$

- This is a linear program, so done!
- Look in detail at what simplex really does.
- Maximize total flow. $|f| = \sum_{(s,v)} f(s,v)$
  - Start with zero flow
  - Choose an "augmenting" path from s to t and increase flow on the edges of that path as much as possible.
- To define augmenting paths, we need notion of the residual graph.
- Given flow f:

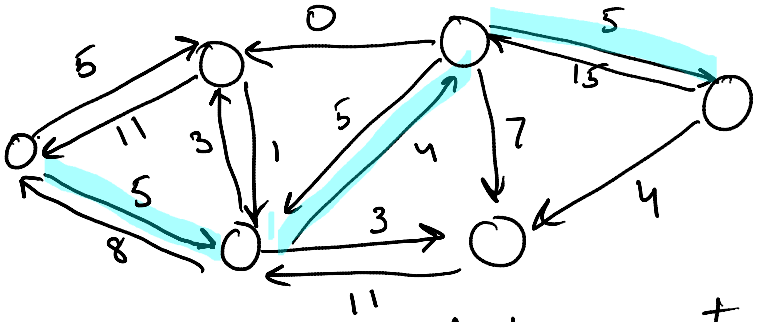
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \end{cases}$$

12/12 ~ 15/-

- Example:



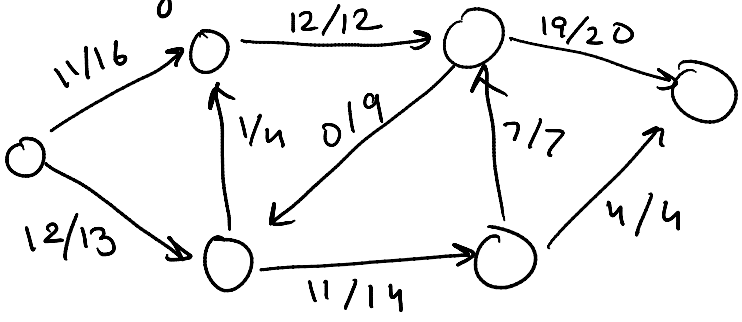
Residual:



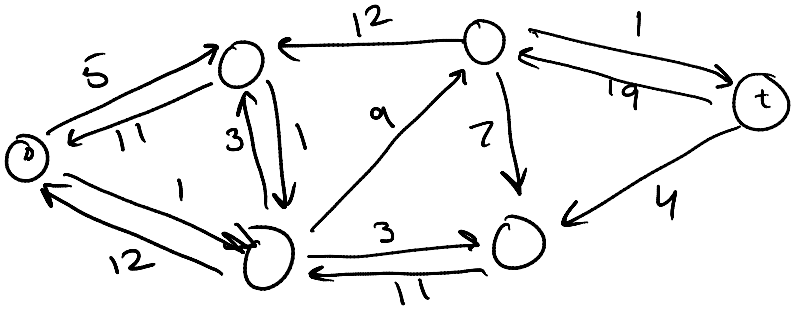
Augmenting path is a directed path from s to t in the residual graph.

Residual capacity of an augmenting path is the least weight edge on the path

- Ford-Fulkerson algorithm:
  Find an augmenting path and increase flow on each edge of the path by the residual capacity of the path.



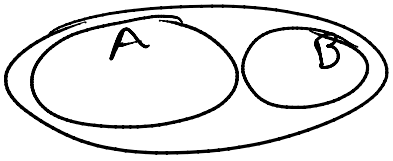
Residual:



No path from s to t.
stop!

Correctness

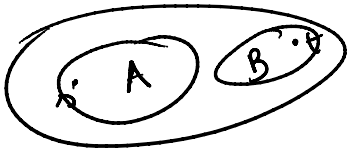
- Ford-Fulkerson always maintains a valid flow.
- Why optimum? We show that if f is the FF flow, then $|f|$ = size of min s-t cut in G.



$$c(A,B) = \sum_{\substack{u \in A, v \in B \\ (u,v) \in E}} c(u,v)$$

$$\text{min } s\text{-}t \text{ cut} = \min_{\substack{A \ni s \\ B \ni t \\ A \cap B = \emptyset}} c(A,B)$$

- Clear that $|f| \leq c(\{s\}, V \setminus \{s\})$
  $\leq$ min s-t cut
- Now when FF stops, s and t disconnected in residual graph.



If $(u,v) \in E$ with $u \in A, v \in B$,
$f(u,v) = c(u,v)$.

If $(v,u) \in E$ with $u \in A, v \in B$
$f(u,v) = 0$.

So, $|f| = c(A,B)$

Max Flow - Min Cut Theorem