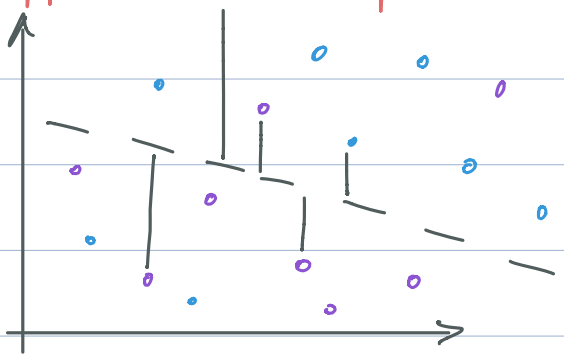
CS124 Lecture 18

Monday, April 6, 2020

Approx Linear Separator



Input: $(x_i, y_i)_{i \in n}$ is blue

$(x_{n+j}, y_{n+j})_{j \in n+1 \text{ to } m}$

ε_i = error for each point → minimize $\sum \varepsilon_i$

$\varepsilon_i \geq 0$

if $y_i \geq ax_i + b$ $\quad \varepsilon_i = 0$

$y_i < ax_i + b$ $\quad \varepsilon_i = ax_i + b - y_i$

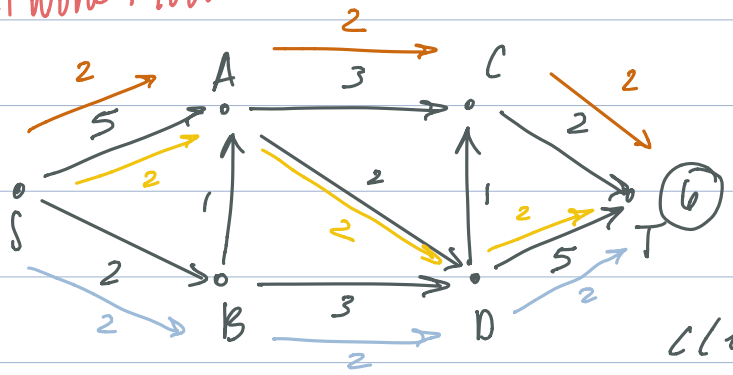
] for blue, which should be above line

$\varepsilon_i \geq (ax_i + b) - y_i$ for blue points

$\varepsilon_i \geq y_i - (ax_i + b)$ for green points

$\varepsilon_i = \max(0, ax_i + b_i - y_i)$

Network Flow



S → T network flow

Source S, sink T

Directed Edges

$c(e)$ = capacity of each edge

Assign value $f(e)$ to each edge

e_i (initially $f(e)$) units of stuff cross edge

Goal: Get as much stuff as possible from S to T.

Linear Program Formulation

var f_e : flow along each edge

$0 \leq f_e \leq c_e$ capacity constraints

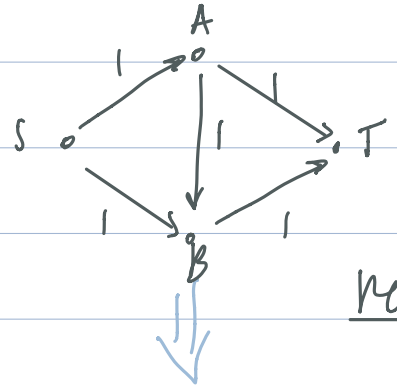
$f_{SA} + f_{BA} = f_{AC} + f_{AD}$ conservation flow constraints

objective:

max $f_{CT} + f_{DT}$ = flow going to T

$\equiv$ max $f_{SA} + f_{SB}$

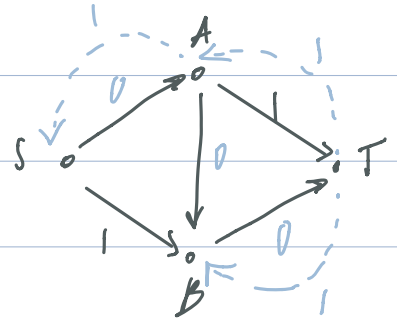
More Examples



max flow 2

S - A - B - T is a wrong decision

Remaining capacity = $c(e) - f_{cur}(e)$



Greedy doesn't allow reverses, so build "backedges"

$C_{AS} = 0$ $C_{SA} = f_{SA} = 1$

$f_{AS} = -1$!

Residual capacity: $C_{AS} - f_{AS} = 1$

Algorithm

Definition: Residual capacity = $c(e) - f(e)$

when $f((u,v)) = -f((v,u))$

Alg. Max S-T Flow

Start w/ residual capacity network = original ~~DFS~~ BFS

Find path (augmenting path) from S to T

push as much flow as possible along path

update flow values, update capacities in residual capacity network

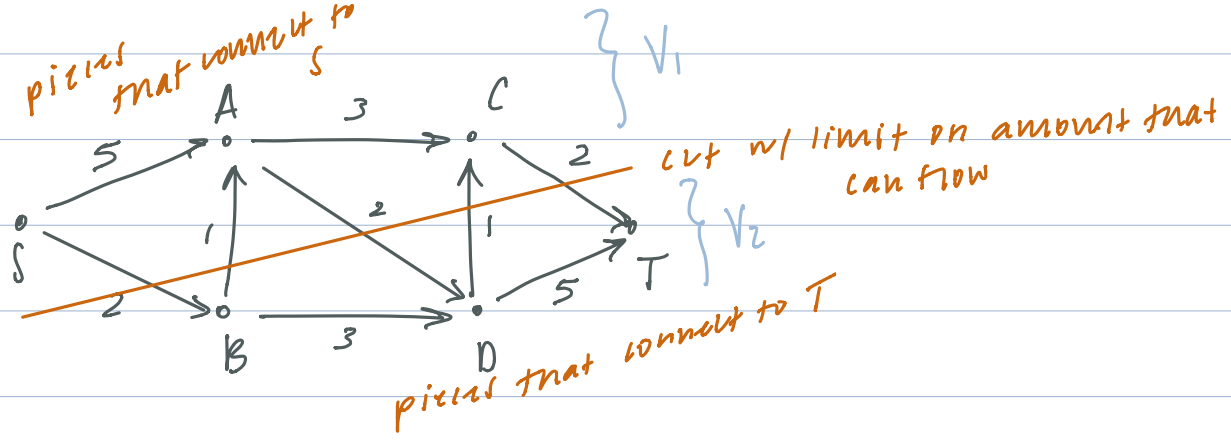
until no more paths are found

Runtime

- Given integer capacities ← if $\in \mathbb{R}$, can be infinite runtime
- DFS is $O(|E|)$
- If max flow is f^*, running time is $O(\text{max flow} \cdot \text{DFS runs})$

Edmonds-Karp Alg

- $O(E^2V)$ w/ capacities $\in \mathbb{R}$
- same algo as above, uses BFS instead of DFS



max flow = min cut!

Cut = partition into two sets V_1, V_2 $\quad S \in V_1, \ T \in V_2$

Capacity of cut = $\sum_{\substack{u \in V_1 \\ v \in V_2}} c((u,v))$

Claim. Maximum flow = minimum cut

↳ sum of edges crossing the cut

Implications.

- Integer capacities → integer solution
  (flow on each edge is an int)

because we push as much flow as possible on paths (by def of the algorithm)

→ integer LP is an NP problem

- LP Duality
  corresponding minimization problem has same solution