Lecture 3.3

Monday, September 02, 2013
9:33 PM

Blocking Flows (Dinic)

Idea: use multiple augmenting paths simultaneously.

Layered graph: By distance from source

Admissible arc: from layer $d-1$ to d

Admissible path: Using only admissible arcs

blocking flow: saturates at least one arc on every admissible path

Lemma: Adding a blocking flow increases source-sink distance in residual network.

Proof: In new residual network, there is no s-t path that only uses admissible arcs from previous residual network. So, every s-t path in new residual network must be longer than shortest path in previous residual network since (s,x) and (x,t) distances are non-decreasing for all $x \in V$.

Corollary: n blocking flows yields a max flow

Q: What is the running time for finding a blocking flow?

ADVANCE: Take an admissible edge in forward direction

RETREAT: If no outgoing edge, go back along the edge deleting it

If you reach the sink, augment along path found and reduce capacity on edges by the amount of flow augmentation. Start from the source again.

Unit capacity graphs: Advance / Retreat only once per edge $\Rightarrow O(m)$ time for blocking flow and $O(mn)$ overall

Arbitrary capacities: Advance at most m times per edge since only one edge deleted per augmenting path $\Rightarrow O(m^2)$ time for blocking flow and $O(m^2 n)$ overall.

Alternative bound for unit capacity graphs

After d iterations, residual maxflow is $\leq m/d$ since it has to be decomposable into paths of length $\geq d$.

$\Rightarrow$ # of iterations $\leq d + m/d$ for any d

Set $d = \sqrt{m}$ to obtain running time bound of $O(m^{3/2})$. This is better than $O(mn)$ since $\sqrt{m} \leq n$.

For capacitated graphs,

- charge advances to augment/retreat
- retreat deletes edges $\Rightarrow O(m)$
- augment deletes at least one

edge $\Rightarrow$ $O(m)$ augments
But, each augment can have
$O(n)$ advances charged to it
since augmenting path may
have upto n edges

$\Rightarrow$ Overall running time $= O(mn)$ per
blocking flow $\Rightarrow O(mn^2)$ for flow algo