

## Lecture 7 Maximum Flow : 2.11.08

*Lecturer: Satish Rao**Scribe: Rao, for now***Disclaimer:** *These are rough notes, with some exercises.***7.1 Maximum flow: should be review.****Question: What is the maximum flow problem?**

Given a directed graph  $G = (V, E)$  with integer capacities,  $c : E$ , and a source node  $s$  and a sink node  $t$  in  $V$ . Find a flow function on the arcs subject to “conservation constraints” and “capacity constraints.” The conservation constraint for a node is that the sum of the flow on incoming arcs is equal to the sum on outgoing arcs for all nodes other than the source and the sink. The capacity constraint for an arc is that the flow is no larger than the capacity.

One wishes to maximize the flow out of the source (into the sink.)

**Question: What is an upper bound on the flow?**

The minimum  $s$ - $t$  cut in the graph.

**Question: Is it tight?**

Yes.

**Question: What are the residual capacities for a flow function on a graph?**

The capacity “left” on each edge. That is,  $c(e) - f(e)$  for each edge in  $G$ , and  $f(e = (i, j))$  for each reverse edge  $(j, i)$ . That is, one could either increase the flow along  $e$ , or decrease it. The latter operation can be seen as pushing flow along the reverse edge of  $e$ .

**Question: What is the Ford-Fulkerson maximum flow algorithm?**

Find an  $s$ - $t$  path consisting of strictly positive residual capacity arcs. The flow will continue to increase until...

**Question: How do we terminate?**

Can no longer find an  $s$ - $t$  path in residual network.

**Question: What does this mean?**

There is a set of nodes that is reachable from  $s$ , call it  $S$ , and the set containing  $t$ ,  $T$ . And all the outgoing arcs from this set are “saturated”, i.e., have zero residual capacity. There is no arc with flow going from  $S$  to  $T$  since it would have residual capacity. Thus, the capacity of the cut is equal to the flow across the cut.

**Question: This is an  $s$ - $t$  cut. What is its size?**

The size of the flow. The sum of the flow in and out of all the nodes is equal to the flow across the cut. By

conservation, only the sink has an excess flowing into it. Thus, the value of the value is equal to the capacity of the arcs going into the set,  $T$ .

## Exercise 1: How long does it take to verify that a solution is maximum?

**Question: Is this optimal?**

Sure. The capacity of an  $s - t$  cut is an upper bound on the flow.

**Question: How long does this take?**

$O(mC)$ , where  $C$  is the total flow value.

**Question: Is this fast?**

Can do quite badly. For example, consider a four node network, which consists of a  $s$ ,  $a$ ,  $b$ , and  $t$ . Capacity 1 arc;  $(a, b)$ . Capacity 1,000,000,000 arcs;  $(s, b)$ ,  $(s, a)$ ,  $(a, t)$ ,  $(b, t)$ .

**Question: Bad sequence?**

Sure, route  $s$  to  $a$  to  $b$  to  $t$ , and then from  $s$  to  $b$  to  $a$  to  $t$ , and then on the first path, and then on the second. This leads to around a 1,000,000,000 augmentations.

**Question: Fix it? Route along shortest paths?**

Sure, this works better.

## 7.2 Edmonds-Karp.

**Question: Always?**

Sure. Always choose shortest augmentation.

**Question: Progress?**

Source to sink distance. Indeed, think of distance to sink of every node, i.e.,  $d(u)$ .

**Question: What does augmenting path look like with respect to these distances?**

It goes along nodes dropping one unit of distance in each step. That is,  $d(u) > d(v)$  for all edges  $(u, v)$  on an augmenting path.

**Question: Distance to sink from any node is nondecreasing?**

The algorithm only adds arcs that are “uphill”, (i.e.,  $(u, v)$ ,  $d(u) < d(v)$ ) since residual capacities only increase on reverse arcs of augmenting path. Thus, no node can get closer to sink; consider the closest to the sink  $v$  where  $d(v)$  decreased, the first arc  $(v, u)$  must be new, otherwise there is closer node, but then the reverse arc  $(u, v)$  was used in the augmentation, and  $d(u) = d(v) + 1$ , and now  $d(u)$  is less than the new (even lower value of)  $d(v)$ . Thus  $d(u)$  must have decreased. This contradicts choice of  $v$ .

Indeed, this shows that any path from  $v$  using a new arc must be longer in some sense than previously existing paths.

**Question: When edge  $(u, v)$  is saturated (i.e., no residual capacity) what must be the case?**

$d(u) > d(v)$ .

**Question: Once edge  $(u, v)$  is saturated, when can it have residual capacity again?**

Only when  $d(v) < d(u)$ . Only then can we create residual capacity. That is, to unsaturate an arc, one must increase the distance to the sink of its tail by two.

**Question: How many times can an edge be saturated?**

At most  $n/2$ , since distance to sink never gets larger than  $n$  without and each saturation can only be undone when the tail rises by 2.

**Question: Can we bound the number of augmentations?**

Sure, each augmentation saturates one edge. The total number of edge saturations is  $O(nm)$ . Thus, the total number of iterations is  $O(nm)$ . The total time is  $O(nm^2)$ .

## 7.3 Dinitz.

**Question: Leveled network. Definition?**

Put nodes into levels according to  $d(v)$ . Remove all arcs that don't go from levels  $l$  to  $l - 1$  for some  $l$ .

**Question: Any arc from level 10 to level 1?**

No. Definition of distance.

**Question: Block this network! What?**

Compute a flow where, for any  $s$ - $t$  path in the leveled network, at least one arc is saturated.

**Question: What happens to  $d(s)$  after blocking flow?**

Grows by 1, since any current shortest  $s$ - $t$  path is no longer feasible, and any other old path is longer by at least one, and any new path uses an "uphill" arc, and thus must be longer (by two).

**Question: How many iterations?**

At most  $n$ .

**Question: How long does it take to compute a blocking flow?**

At most  $O(nm)$ , use depth first search and augment along a path, and restart depth first search. All the work can be charged to the push/pops of the depth first search. Any pop during any depth first search removes an edge forever (we remove it from the graph for future dfs since it does not lead to  $t$ .) An augmentation may pop  $n$  nodes but removes at least one edge, and thus only happens  $m$  times. Thus total is  $O(nm)$ .

**Question: Faster?**

For each node consider the minimum of the incoming capacity and the outgoing capacity. Consider the minimum over all nodes.

Route this amount of flow from minimum node to sink. Can always continue pushing to sink since every intermediate node has strictly greater out capacity. Repeat until no nodes left.

(This is an algorithm of Karzonov.)

**Question: For an intermediate node, if it has outdegree  $d$  in one iteration. Can you say something about first  $d - 1$  arcs used?**

The first  $d - 1$  arcs are “saturated”, the last may not be. Thus, one can remove the first  $d - 1$  arcs. Thus the total run time for an iteration  $O(n)$  plus  $O(\text{the number of saturations})$ .

(Push back in the same manner.) Repeat until all nodes are handled. Total time is  $O(n^2)$  plus  $O(\text{the total number of saturations})$  which is  $O(m)$  since only  $m$  edges.

**Question: Total time for max flow?**

$O(n^3)$ .

### 7.3.1 Matching: review

**Question: Consider the maximum matching problem. That is given a bipartite graph  $G$  find a maximum set of edges that share no endpoints. One side could be jobs, the other workers; choose your own example. Can you solve this problem?**

Reduce it to the maximum flow problem. Add a source, and connect it to each node on one side, say the left side. Add a sink to and draw unit arcs from the right side to the sink. Direct the arcs in the middle from left to right.

**Question: Given a matching of size  $m$ , is there a flow?**

For each matching edge, route a unit of flow from the source to the left endpoint, across the matching edge, and then to the sink.

**Question: Given a flow of value  $m$ , is there a matching?**

Hmmm...what if we have  $1/2$  everywhere.

**Question: Is the optimal flow, integral?**

Sure. All the algorithms above only route integral flows. They are optimal as well, so there exists an optimal integral flow.

**Question: Given an integral flow of value  $m$ , is there a matching?**

Sure. Choose arcs with unit flow in middle level. Only one arc leaves any left node, or enters any right node. There are  $m$  of them.

### 7.3.2 Matching:algorithm.

**Question: Can we do better than  $O(n^3)$  time? Recall that  $n$  blocking flows is sufficient. Can we do blocking flows faster than  $O(n^2)$  in this network?**

Sure,  $O(m)$  by depth first search. Again, you push an edge, pop an edge, or augment. In the case that you augment, all the edges on the stack (or on the path) will be saturated and removed from the residual network.

**Question: Actually, even simpler, how much flow is there to route?**

At most  $n$ . Each augmentation is  $O(m)$  time.  $O(nm)$ .

**Question: Can we do even better? In particular, can we run fewer than  $\Theta(n)$  blocking flow computations?**

Maybe.

**Question:** Once the depth of the network is  $l + 1$ , is there a level with few nodes?

Sure, there is a level with at most  $n/l$  nodes in it since there are  $l$  levels between  $s$  and  $t$  and the total number of nodes contained in those levels is at most  $n - 2$ .

**Question:** Can you use this small level to produce a cut in the residual network?

Each of these nodes either has indegree at most 1 or outdegree at most 1. Thus, there is a cut of size  $n/l$ .

**Question:** Does this upper bound the remaining flow?

Thus, the flow remaining after  $l$  blocking flows is  $O(n/l)$ .

**Question:** So how many augmenting paths do we need to find?

In particular, after  $\sqrt{n}$  augmentations one only needs to do  $\sqrt{n}$  augmentations to finish. This gives a time of  $O(\sqrt{nm})$ .

**Exercise 2:** Give the best upper bound on the total length of the augmenting paths in the bipartite matching problem above.

**Exercise 3:** Give an  $O(m^{3/2})$  time algorithm for finding maximum flows in a network with unit capacity edges. (Note, this is more general than the flow network corresponding to the bipartite graph model.) Give an argument justifying this runtime bound.

**Exercise 4:** Give an  $O(n^{2/3}m)$  time algorithm for finding maximum flows in a network with unit capacity edges with no parallel arcs. Give an argument justifying this runtime bound.