ORIE 633 Network Flows October 2, 2007

Lecture 9

Lecturer: David P. Williamson *Scribe: Qiu Wang*

1 Efficient algorithms for max flow

1.1 Blocking flow and Dinic's algorithm

In this lecture, we'll discuss about some other efficient algorithms for computing a maximum flow. They are based on the concept of a *blocking flow*.

Definition 1 *A flow f in G is* **blocking** *if every s-t path in G has some arc saturated.*

Every maximum flow is obviously also a blocking flow. Is every blocking flow a maximum flow? No, Figure 1 is a counterexample.

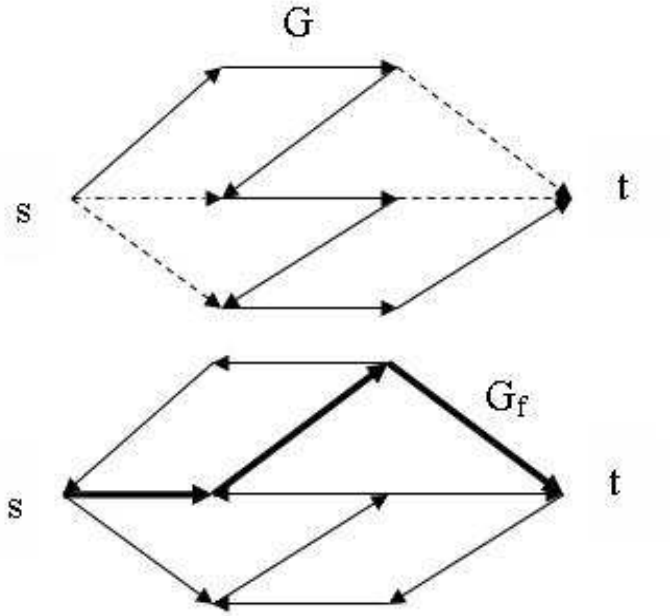

Figure 1: A blocking flow is not necessary a max flow. Solid arrows in the top figure show an augmenting path that gives a blocking flow. Bold arrows in the residual graph show another augmenting path.

As shown in Figure 1, there exists an s-t path in the residual graph, which implies that the current flow is not maximum.

Let d_i be the distance of node i to sink t.

Definition 2 *An arc is called* **admissible** *if* $(i,j) \in A_f$ *and* $d_i = d_j + 1$.

Note that an admissible arc is on some shortest path to t. We now turn to an algorithm that use blocking flows as a subroutine.

Dinic's Algorithm (Dinic 1970)

$f \leftarrow 0$
while $\exists s$-t path in G_f
    Compute distances d_i to sink t in G_f $\forall i \in V$
    Find blocking flow $\tilde{f}$ in graph $\tilde{G}$ with arcs $\tilde{A} = \{(i,j) \in A_f : d_i = d_j + 1\}$
        capacity u_{ij}^f
    $f \leftarrow f + \tilde{f}$.

In the problem set, we considered an augmenting path algorithm in which we sent flow down the shortest path in the residual graph each time. In Dinic's algorithm we effectively saturate all the shortest paths at the same time.

The running time of the algorithm depends on how fast we can find a blocking flow. The following theorem shows what we can do so relatively quickly.

Theorem 1 *Blocking flows in acyclic graphs can be found in* $O(mn)$ *time, but if fancy data structures are used, then they can be found in* $O(m \log n)$ *time.*

The easier part of the theorem is a problem on the current problem set; the harder part will be a bonus problem on a later problem set.

Note that the efficient algorithms for computing blocking flows are for *acyclic* graphs. The set of admissible arcs is acyclic, otherwise we would have an inconsistency of the shortest path distances $d_i = d_j + 1$; we cannot have a cycle of edges all of which are on a shortest path to the sink.

The following lemma is the key to proving a bound on the running time of Dinic's algorithm.

Lemma 2 *The distance to the sink* d_s *strictly increases in each iteration of the algorithm.*

Clearly this implies that the algorithm takes at most n iterations. Given the two blocking flow algorithms mentioned above, we get the following results.

Theorem 3 *Dinic's algorithm can be implemented in* $O(mn^2)$ *time (Dinic 1970), or* $O(mn \log n)$ *time (Sleater, Tarjan 1980).*

Now we turn to the proof of the lemma.

Proof of Lemma 2: Let d_i be distance labels in one iteration, d'_i in the next iteration. Let f be the flow in one iteration, f' in the next iteration.

To begin, we claim that the d_i is a valid distance labeling for the flow in the next iteration; that is, $d_i \leq d_j + 1$ for all $(i,j) \in A_{f'}$. Why? How does some arc come to exist in $G_{f'}$? Only two cases can cause this to happen:

- $(i, j) \in A_f$. In this case, we have $d_i \leq d_j + 1$, by the definition of shortest paths.
- We sent flow on arc (j, i). In this case, we have $(j, i) \in A_f$ and $d_j = d_i + 1$. This implies $d_i = d_j - 1 \leq d_j + 1$.

Thus the claim holds.

We want to show that $d'_s > d_s$. Look at any s-t path P in $A_{f'}$. By the properties of a blocking flow, there exists an arc $(i, j) \in P$ that was not admissible in the previous iteration; $(i, j) \notin \tilde{A}$. In other words, either $(i, j) \notin A_f$, or $d_i \neq d_j + 1$. In the former case, we must have had flow sent on (j, i) in the previous iteration, which as above implies that $d_i < d_j$. The latter case implies that $d_i \leq d_j$ since $d_i \leq d_j + 1$. Since d_i is a distance labelling for the arcs in $A_{f'}$, this implies that $|P| > d_s$, which implies that $d'_s > d_s$. The reason for this can also be seen in Figure 2. Recall our definition of a *distance level* $D_k = \{i \in V : d_i = k\}$. After one iteration, any path you take will have to use an arc that stays at the same distance

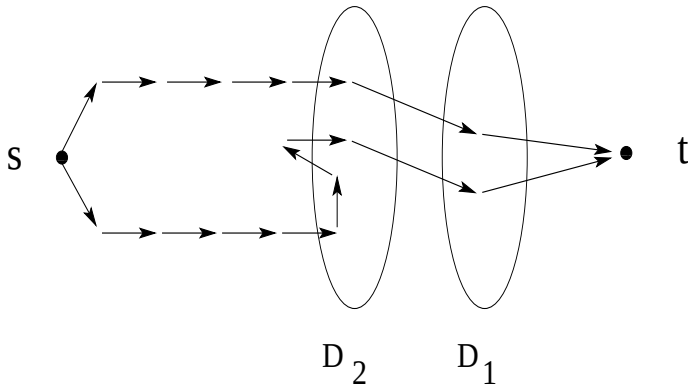

Figure 2: s-t paths with respect to distances d_i before and after a blocking flow.

level or goes backwards with respect to the distances d_i; this implies that the distance from the source to the sink must be larger than d_s. □

1.2 Unit capacity graphs

In some cases, we can show that blocking flow algorithms give a much better result. We consider the special case of *unit capacity* graphs.

Definition 3 *A unit capacity graph has $u_{ij} \in \{0, 1\}$ for all arcs $(i, j) \in A$.*

In this case, we can give the following result.

Lemma 4 *If graph is unit capacity, then Dinic's algorithm takes $O(\min(m^{\frac{1}{2}}, n^{\frac{2}{3}}))$ iterations.*

Proof: We define distance level k as $D_k = i \in V : d_i = k$, and of s-t cuts $S_k = \{i : d_i \geq k\}$; note that for $k > 0$, $s \in S_k$ and $t \notin S_k$.

Suppose first that $d_s \geq m^{\frac{1}{2}}$. Then there exists a distance level D_k such that there are at most $m^{\frac{1}{2}}$ arcs in S_k. As can be seen in Figure 3, arcs from D_k to D_{k-1} are disjoint for all

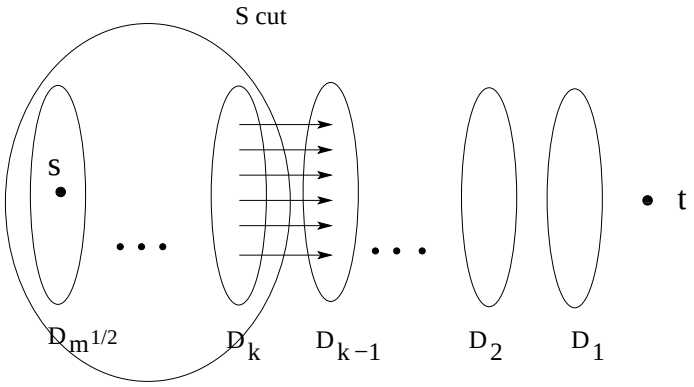

Figure 3: suppose $d_s \geq m^{\frac{1}{2}}$

available distance levels and there are at most m arcs. This implies that if there are at least $m^{\frac{1}{2}}$ distance levels, then there exists a D_k such that there are at most $m^{\frac{1}{2}}$ arcs from D_k to D_{k-1}. Therefore, the residual capacity of the cut S_k is at most $m^{\frac{1}{2}}$ since the graph is unit capacity (that is, $u^f(\delta^+(S_k)) \leq m^{\frac{1}{2}}$). Thus we know that only $m^{\frac{1}{2}}$ more augmentations will be required until the algorithm finds a maximum flow. The algorithm takes $\sqrt{m}$ iterations until the distance from the source is $d_s \geq m^{\frac{1}{2}}$, and $\sqrt{m}$ more iterations until the flow is maximum, for a total of $O(\sqrt{m})$ iterations.

Let us now suppose that $d_s \geq 2n^{\frac{2}{3}}$. Then, there exists D_k, D_{k-1} such that $|D_k| \leq n^{\frac{1}{3}}$ and $|D_{k-1}| \leq n^{\frac{1}{3}}$. To see this, if more than $n^{\frac{2}{3}}$ levels have more than $n^{\frac{1}{3}}$ nodes, we then have more than n nodes in total and contradiction exists. Thus it must be that less than $n^{\frac{2}{3}}$ levels have more than $n^{\frac{1}{3}}$ nodes. This implies at least 2 consecutive levels have less than or equal to $n^{\frac{1}{3}}$ nodes, as shown in Figure 4.

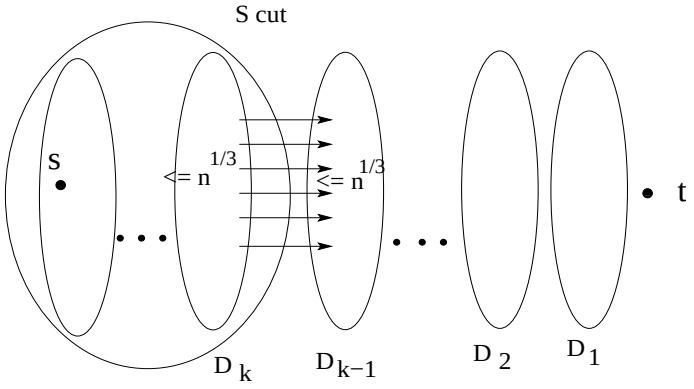

Figure 4: suppose $d_s \geq 2n^{\frac{2}{3}}$

If we now consider all possible arcs from D_k to D_{k-1} (as in Figure 4), there can be at most $n^{\frac{2}{3}}$ arcs. Thus the residual capacity of the cut S_k is $u^f(\delta^+(S_k)) \leq n^{\frac{2}{3}}$, since all arcs have unit capacity. This proves that only $n^{\frac{2}{3}}$ more iterations are needed to find the maximum flow. Thus as in the previous case, after $2n^{\frac{2}{3}}$ iterations, $d_s \geq 2n^{\frac{2}{3}}$ and after $n^{\frac{2}{3}}$ iterations, the maximum flow is achieved, for a total of $O(n^{\frac{2}{3}})$ iterations. □

We claim that in unit capacity graphs, it is easy to find a blocking flow.

Claim 5 *We can find a blocking flow in unit capacity graph in $O(m)$ time.*

This follows since in every s-t path we found can saturate all arcs in the path due to unit capacity, and remove them from further consideration.

Because the quantities in the proof above will come up so frequently in following lectures, let's set aside a special symbol for them.

Definition 4

$$\Lambda = \min(m^{\frac{1}{2}}, 2n^{\frac{2}{3}}).$$

Thus combining the above, we obtain the following.

Theorem 6 *In unit capacity graphs, the maximum flow can be found in $O(\Lambda m)$ time.*

1.3 The Goldberg-Rao algorithm

Now will consider how to apply the ideas of this algorithm to graphs with general capacities (Goldberg, Rao 1998). It takes $O(\Lambda(m \log n)(\log mU))$ operations, where U represent the largest capacity of all edges. We know after Λ blocking flows operations, there exists a cut in the residual graph of capacity at most ΛU, by the same arguments as we used for the

Suppose we can somehow make sure that the arcs from D_k to D_{k-1} have residual capacity at most Δ. Then we know that after Λ iterations, we know (by the proof above) that the remaining residual capacity is $\Lambda\Delta$. This somehow seems useful. How can we obtain such a property? The basic idea we will consider is that of altering the distance function. Up until now, the distance of a vertex to the sink has been the number of arcs on the shortest path. But of course we could have general lengths on the arcs. If we change the length of each arc to be the following:

$$l_{ij} \leftarrow \begin{cases} 1 & \text{if } u_{ij}^f < \Delta \\ 0 & \text{otherwise} \end{cases}$$

then we will get the property that we want, namely, the arcs from D_k to D_{k-1} will have residual capacity at most Δ. In the next lecture we will see how this idea plays out.