ORIE 633 Network Flows August 30, 2007

Lecture 3

Lecturer: David P. Williamson *Scribe: Gema Plaza-Martínez*

1 Polynomial-time algorithms for the maximum flow problem

1.1 Introduction

Let's turn now to considering polynomial-time algorithms for computing a maximum flow. We'll need the following definitions:

Definition 1 *An algorithm runs in polynomial time if the number of operations (basic arithmetic operations, comparisons, etc.) can be bounded by a polynomial in the size of the input, when data items are encoded in binary.*

Definition 2 *An algorithm runs in pseudo polynomial time if the number of operations can be bounded by a polynomial in the size of the input, when data items are encoded in unary.*

Definition 3 *An algorithm runs in strongly polynomial time if the number of operations can be bounded by a polynomial in the number of data items, and it is not dependent on the size of the data items.*

Recall the problem:

Maximum s-t Flow Problem

- **Input:**
  - Directed graph $G = (V, A)$
  - Capacities $u_{ij} \geq 0,\ \forall (i,j) \in A,\ u_{ij}$ integer
  - Source node $s \in V$, sink node $t \in V, s \neq t$
- **Goal:** Find an s-t flow of maximum value.

Definition 4 *An s-t flow is a function $f : A \to \mathbb{R}_+$ such that*

(i) $f_{ij} \leq u_{ij}, \quad \forall (i,j) \in A$ *(capacity constraints)*

(ii) $f_{ij} = -f_{ji}, \quad \forall (i,j) \in A$ *(antisymmetry)*

(iii) $\sum_{k:(i,k)\in A} f_{ik} = 0, \quad \forall i \in V, i \neq s,t$ *(flow conservation constraints)*

Definition 5 *The value of a flow f is $|f| \equiv \sum_{k:(s,k)\in A} f_{sk}$.*

1.2 Augmenting path algorithms

Recall the following definitions:

Definition 6 *The residual graph of a flow f is given by $G_f = (V, A_f, u^f)$, where $A_f = \{(i,j) \in A : f_{ij} < u_{ij}\}$ and $u^f_{ij} = u_{ij} - f_{ij}$.*

Definition 7 *A directed s-t path in G_f is called an augmenting path.*

We describe now the augmenting path algorithm:

Augmenting path

```
$f \leftarrow 0$
While $\exists$ an augmenting path $P$ in $G_f$
    Augment flow on $P$
    Update $f$: $f_{ij} = \begin{cases} f_{ij} + \delta, & \text{if } (i,j) \in P; \\ f_{ij} - \delta, & \text{if } (j,i) \in P; \\ f_{ij}, & \text{otherwise,} \end{cases}$
        where $\delta = \min_{(i,j) \in P} u^f_{ij}$.
```

Let $m = |A|,\ n = |V|$.

Lemma 1 *(Decomposition lemma) Given an s-t flow f, there exists a set $\mathcal{P}$ of s-t paths, a set $\mathcal{C}$ of cycles, and weights $w : \mathcal{P} \cup \mathcal{C} \to \mathbb{R}_+$ such that*

(i) $f_{ij} = \sum_{P \in \mathcal{P} \cup \mathcal{C} : (i,j) \in P} w(P),\ \forall (i,j) \in A$ *s.t.* $f_{ij} > 0$,

(ii) $|f| = \sum_{p \in \mathcal{P}} w(P)$,

(iii) $|\mathcal{P}| + |\mathcal{C}| \leq m$.

Proof: Start with $\mathcal{P} = \mathcal{C} = \emptyset$.

Pick $(i,j) \in A$ with $f_{ij} > 0$. If such an arc doesn't exist, we are done. Otherwise, if $i \neq s$ there exists a k such that $f_{ki} > 0$. This follows from the flow conservation and from the fact that $f_{ij} > 0$.

Similarly, if $j \neq t$ there exists an h such that $f_{jh} > 0$.

Keep repeating this until we find either an s-t path or a cycle. Call this P, and set $w(P) = \min_{(i,j) \in P} f_{ij}$. Add P to $\mathcal{P}$ or $\mathcal{C}$.

Update the flow:

$$f_{ij} = \begin{cases} f_{ij} - w(P), & \text{if } (i,j) \in P; \\ f_{ij} + w(P), & \text{if } (j,i) \in P; \\ f_{ij}, & \text{ow.} \end{cases}$$

Once we are done with this, parts (i) and (ii) follow from the construction. Also notice that at least one more arc has $f_{ij} = 0$, so we can only do it at most m times. Hence (iii) holds. $\square$

Lemma 2 *Let f be an s-t flow, and f^* a maximum flow in G. Then the maximum flow in G_f has value $|f^*| - |f|$.*

Proof: Let f' be an s-t flow in G_f, and define $\widetilde{f}_{ij} = f_{ij} + f'_{ij}$, $\forall (i,j) \in A$. Then $\widetilde{f}$ is a feasible flow in G. This implies $|\widetilde{f}| \le |f^*|$, and therefore $|f'| \le |f^*| - |f|$.

Now, define $\widehat{f}$ as the following flow in G_f: $\widehat{f}_{ij} = f^*_{ij} - f_{ij}$, $\forall (i,j) \in A_f$. Then $\widehat{f}_{ij} = f^*_{ij} - f_{ij} \le u_{ij} - f_{ij} = u^f_{ij}$. So $\widehat{f}$ is a feasible flow and it has value $|f^*| - |f|$. Therefore $\widehat{f}$ is a maximum flow in G_f. $\square$

As we will see in the problem set, this does not necessarily lead to a polynomial-time algorithm because it is possible that there will be too little progress made for each augmentation. However, if one picks a path to make substantial progress in each augmentation, then we can give a polynomial-time algorithm. One natural choice is to pick P with largest possible capacity; i.e. $\max_P \min_{(i,j)\in P}\{u^f_{ij}\}$. Then the algorithm above becomes:

Maximum capacity augmenting path

```
$f \leftarrow 0$
While $\exists$ an augmenting path in $G_f$
    Pick augmenting path $P$ with maximal residual capacity.
    Augment flow on $P$
    Update $f$.
```

The above two lemmas indicate that some augmenting path P will have capacity at least $\frac{|f^*|-|f|}{m}$.

Claim 3 *P has capacity at least $\frac{|f^*|-|f|}{m}$.*

Proof: Follows from the two lemmas. From the second lemma, the maximum flow value in G_f is $|f^*| - |f|$. By the decomposition lemma, there are at most m paths in G_f on which we can send $|f^*| - |f|$ in total. $\square$

Let's consider $2m$ iterations of the loop in the above algorithm. Let f^0 be the flow at the beginning of them. Then either

(i) all $2m$ iterations augment flow value by $\ge \frac{|f^*|-|f^0|}{2m}$. Or

(ii) at least one iteration augments flow by $< \frac{|f^*|-|f^0|}{2m}$.

If (i) happens, we are done, since we have a flow of value at least $|f^*|$. If (ii) happens, then the capacity of the maximum capacity augmenting path has dropped by a factor of 2. The upper bound on the capacity of P is $U \equiv \max_{(i,j)\in A} u_{ij}$. The lower bound on the capacity of P is 1. Therefore, there can be at most $O(\log U)$ decreases of the capacity of the maximum capacity augmenting path by a factor of 2. Since every $2m$ iterations either the algorithm terminates with a maximum flow or the capacity drops by a factor of 2, there are at most $O(m \log U)$ iterations of the main loop overall. This gives a polynomial-time algorithm.

To get the exact running time, we would have to determine the time needed to find the maximum capacity augmenting path. Rather than get into this, we will consider a variation of the algorithm above in which we only need to find a path in a network. The idea of this algorithm is to look for paths in which each edge is 'big'. If such a path exists, then we can increase the flow by a significant amount. If there is no such path, then we will show that we must be closer to the maximum flow value. We first need the following definition.

Definition 8 *A δ-capacity augmenting path is an augmenting path P such that $u^f_{ij} \geq \delta$ for all $(i,j) \in P$.*

The algorithm is as follows:

Capacity scaling

```
$f \leftarrow 0$
$\delta \leftarrow 2^{\lfloor \log_2 U \rfloor}$
While there is an augmenting path in $G_f$
    If $\exists$ $\delta$-capacity augmenting path $P$
        Augment flow on $P$, update $f$.
    Else
        $\delta \leftarrow \delta/2$
```

Suppose that there is no δ-capacity augmenting path and we decrease δ by 2. Let δ' be the new value and δ the old value. Then the maximum flow value in G_f is $< m\delta = 2m\delta'$ by the decomposition lemma. This implies that we can't find more than $2m$ δ'-capacity paths before we have to halve δ' again. This means at most $O(m \log U)$ iterations.

The disadvantage of augmenting path-style algorithms is that we have to find the full path each time, and this takes $O(m)$ time. We are going to see another type of algorithm that avoids this.

1.3 The push/relabel algorithm

Observation 1 *A pre-flow doesn't satisfy the flow conservation constraints (iii), but it has $\sum_{(j,i)\in A} f_{ji} \geq 0, \; \forall i \in V, i \neq s, t$.*

Definition 9 *A distance labelling is a set of d_i for all $i \in V$ such that:*

- *d_i is a non-negative integer;*
- *$d_t = 0$;*
- *$d_s = n$;*

- $d_i \leq d_j + 1 \quad \forall\, (i,j) \in A_f$.

The idea of distance labelling is the following:

- If $d_i < n \;\Rightarrow$ it gives a lower bound on the distance to t.
- If $d_i \geq n \;\Rightarrow$ it gives a lower bound on the distance to s.

Definition 10 *If $e_i > 0, i \in V, i \neq s, t$ then we call i active.*

Push/Relabel(Goldberg, Tarjan '88)

```
$f \leftarrow 0$
$f_{sj} \leftarrow u_{sj}, \quad f_{js} \leftarrow -u_{sj}, \quad e_j = u_{sj}, \quad \forall (s,j) \in A$
$d_s \leftarrow n$
$d_i \leftarrow$ distance to $t$ in $G$, $\forall i \in V, i \neq s$
While $\exists$ active $i$
    If $\exists j$, s.t. $u^f_{ij} > 0$ and $d_i = d_j + 1$
        Push $\delta \leftarrow \min(e_i, u^f_{ij})$
        $f_{ij} \leftarrow f_{ij} + \delta; \quad f_{ji} \leftarrow f_{ji} - \delta$
        $e_i \leftarrow e_i - \delta; \quad e_j \leftarrow e_j + \delta$
    Else
        Relabel $d_i \leftarrow \min_{(i,j) \in A_f} (d_j + 1)$.
```