ORIE 633 Network Flows September 20, 2007

Lecture 6

Lecturer: David P. Williamson *Scribe: Animashree Anandkumar*

1 Polynomial-time algorithms for the global min-cut problem

1.1 The global min-cut problem

Recall from the previous lecture the global min-cut problem and the claim:

Global Min-cut

- **Input:**
  - directed graph G=(V,A)
  - capacities $\nu_{ij} \geq 0 \ \forall (i,j) \in A$, *integer*
- **Goal:** Find $S \subset V, S \neq \emptyset$ that minimizes $\nu(\delta^+(S))$

Definition 1 *Min s-t cut: Input $s,t \in V$. Find $S : s \in S, t \notin S$ that minimizes $u(\delta^+(S))$.*

Definition 2 *Min s-cut: Input $s,t \in V$. Find $S : s \in S$ that minimizes $u(\delta^+(S))$.*

We showed last time the following two lemmas.

Lemma 1 *We can find the global min-cut by running a min s-cut algorithm twice.*

Lemma 2 *We can find the min s-cut with $n-1$ max flows.*

In fact, we can also find the min s-cut by finding something more exotic, called a minimum X-t cut.

Definition 3 *The min X-t cut: Input $X \subset V, X \neq \emptyset, t \in V - X$. Find $S : X \subseteq S, t \notin S$ that minimizes $u(\delta^+(S))$.*

To find min s-cut using min X-t cut algorithm

$X \leftarrow \{s\}$, $cutval \leftarrow \infty$, $cut \leftarrow \emptyset$
While $X \neq V$
    Pick any vertex in $V - X$ as t
    Find S : min X-t cut
    If $u(\delta^+(S)) < cutval$
        $cutval \leftarrow u(\delta^+(S))$ and $cut \leftarrow S$
    $X \leftarrow X \cup t$
Return *cutval*, *cut*

Claim 3 *The above algorithm finds the min s-cut.*

Proof: Let S^* be a min s-cut. Consider the first iteration s.t. some t is chosen with $t \notin S^*$. In that iteration, $X \subseteq S^*$ and $t \notin S^*$. So S^* is an X-t cut and hence, the min X-t cut found in that iteration can have capacity at most $u(\delta^+(S^*))$. Also, since any X-t cut is an s-cut, its capacity should be at least $u(\delta^+(S^*))$. Hence, the min X-t found at the end of this iteration has the value $u(\delta^+(S^*))$ and is a min s-cut. □

We will now show how we can implement the algorithm given in the proof above to find a minimum s-cut. First we need a few definitions.

Definition 4 *A* distance level k, D_k, *is the set* $\{i \in V : d_i = k\}$.

Definition 5 *Distance level k is* empty *if* $D_k = \emptyset$.

Definition 6 *Distance level k is called a* cut level *if* $|D_k| = 1$ *and for* $i \in D_k, \forall (i,j) \in A_f, d_i < d_j$.

We will now establish why cut levels are useful when finding min cuts.

Lemma 4 *If the distance level k is a cut level, then for* $S = \{i : d_i \geq k\}$, *all arcs in* $\delta^+(S)$ *are saturated.*

Proof: Pick any $(i,j) \in \delta^+(S)$. By definition of S, $d_i \geq k, d_j < k$. If $d_i = k$, then by definition of cut level $(i,j) \notin A_f$ which implies that (i,j) is saturated. If $d_i > k$, then $d_i > d_j + 1$, and thus $(i,j) \notin A_f$, which again implies that (i,j) is saturated. □

The intuition is that the min cut can be found when there are no active nodes strictly below the cut level. Here is the implementation of the push/relabel algorithm to find the min s-cut. Note that this is just a more detailed implementation of the algorithm above, so that if this algorithm in fact finds the minimum X-t cut in every iteration, it will find the minimum s-cut.

Push/relabel min s-cut (Hao and Orlin, 1993)

$X \leftarrow \{s\}$; Pick any vertex in $V - X$ as t
$d_s \leftarrow n, d_t \leftarrow 0, d_i \leftarrow 0, \forall i \in V - \{s\}$
$cutval \leftarrow \infty$, $cut \leftarrow \emptyset$
While $X \neq V$
  Run Push/Relabel which selects only active nodes i with $d_i < k$
    for lowest cut level k (or $d_i < n-1$ if no cut level)
  Let k be lowest cut level ($n-1$ if no cut level)
    Note that there are no active nodes i with $d_i < k$.
  $S \leftarrow \{i : d_i \geq k\}$
    S is the min X-t cut.
  If $u(\delta^+(S)) < cutval$
    $cutval \leftarrow u(\delta^+(S))$ and $cut \leftarrow S$
  Pick $t' \neq t : d_t \leq d_i, \forall i \in V - X - \{t\}$
  Let $X \leftarrow X \bigcup \{t\}$
  Let $d_t \leftarrow n$ and saturate all arcs out of t
  Set $t \leftarrow t'$
Return *cutval*, *cut*.

We will show the following lemmas.

Lemma 5 *If $i \notin X, d_i \leq n-2$.*

Lemma 6 *Each time through the while loop, S is the min X-t cut.*

Before we prove these two lemmas let us first take a look at their implications. We will argue below that these next lemmas follow from Lemma 5.

Lemma 7 *There are at most $O(n^2)$ relabels.*

Lemma 8 *There are at most $O(nm)$ saturating pushes.*

Lemma 9 *There are at most $O(n^3)$ non-saturating pushes.*

Lemma 7 is true since each time a node other than the sink is relabelled, its distance label increases by at least 1 and the total increase is bounded by $n-2$ (by Lemma 5). The distance label of the sink is set to n at the end of the iteration and it is not relabelled further. Lemma 8 holds since between 2 saturating pushes on an arc, the distance labels of its end nodes must have increased by 2. Again, as the distance labels are bounded, the number of saturating pushes is $O(n)$ for any arc and $O(mn)$ overall. Lemma 9 can be shown using the FIFO implementation of the push/relabel algorithm and a modified potential function.

Recall that we also showed last time that two executions of a min s-cut algorithm can be used to find a global min cut. Then the above lemmas lead to the following theorem.

Theorem 10 *(Hao, Orlin 1993) The push/relabel min s-cut finds the min s-cut and global min-cut in $O(n^3)$ time.*

In fact, it can be shown that algorithm can run in $O(mn \log n)$ time. We contrast this running time with the earlier "crude" estimates of $(n-1)$ and $n(n-1)$ max flow computations required to find a min s-cut and global min-cut, respectively.

Now, we return to the proofs of Lemmas 5 and 6. We start with Lemma 6.

Proof of Lemma 6: We know that $d_i = n \; \forall i \in X$, $d_i \leq n-2 \; \forall i \notin X$ and the sink t has the minimum distance label. Now, the way in which S is chosen implies that $X \subseteq S$ while $t \notin S$. Also, since any node with an excess is inside S and all arcs in $\delta^+(S)$ are saturated, S is a min X-t cut. □

To prove Lemma 5, we first need the following lemma.

Lemma 11 *The non-empty distance levels k for $k < n$ are consecutive.*

Proof: This is clearly true at the beginning of the algorithm. If some distance level D_l with $l < n$ becomes empty, let i be the last node in D_l. We will consider two cases for i to leave D_l.

- Case (1): i is relabelled. This happens if $e_i > 0$ and $d_i < k$ for lowest cut level k. Consequently, $d_i \leq d_j \; \forall (i,j) \in A_f$, since i needed to be relabelled. This is a contradiction since $|D_l| = 1$, implying that l is the lowest cut level, not k.
- Case (2): i is sink t and $d_i \leftarrow n$ at the end of the execution of that loop. Then, in the previous iteration i had the minimum distance d_i. Also, since i is a sink, its distance has not increased. Therefore, i is still the minimum d_i and setting d_i to n does not contradict the lemma.

□

Now we can prove Lemma 5.

Proof of Lemma 5: By induction on $|X|$. Let $i \notin X$ have the max distance label. Noting that at each iteration the current sink t has the lowest distance label, Lemma 11 implies that the distance levels between d_t and d_i are all non-empty.

Initially, $X = \{s\}$, $d_t = 0$, and since each distance level between d_i and d_t must contain at least one vertex from the remaining $n - |X| - 1$ vertices,

$$d_i \leq d_t + (n - |X| - 1) \leq n - 2$$

Now assume that $d_i \leq d_t + (n - |X| - 1) \leq n - 2$ at the start of an iteration. At the end of this iteration, $|X|$ increases by 1 as the current sink t is added to X. The distance label of the new sink t' increases by at most 1. This is because even if the lowest distance level becomes empty after t has been added to X, there must be a node in the next higher distance level (by the property that the non-empty distance levels are consecutive). Letting d' denote the distance labels in the next iteration and $X' = X \cup \{t\}$,

$$d'_i \leq d'_t + (n - |X'| - 1) \leq d_t + 1 + (n - (|X| + 1) - 1) \leq n - 2.$$

□

So, in directed graphs an algorithm for finding a global min-cut is based on a max-flow computation. Next time, we will look at algorithms for finding a global min-cut in undirected graphs which seem to have nothing to do with flows.

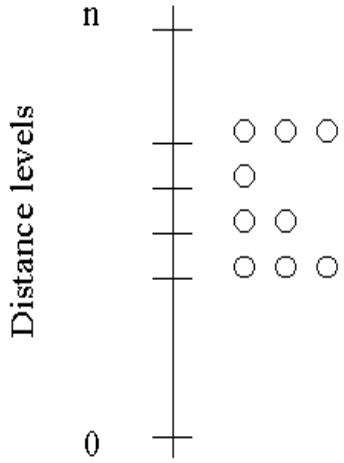

Figure 1: Consecutive non-empty distance levels.