ORIE 633 Network Flows	September 27, 2007
Lecture 8	
Lecturer: David P. Williamson	*Scribe: Gema Plaza-Martínez*

1 Global min-cuts in undirected graphs

1.1 Random contraction

Recall from last time we introduced the random contraction algorithm.

Random Contraction (Karger '93)

Until $|V| = 2$
  Pick edge (i, j) at random with probability proportional to its capacity.
  Contract (i, j).

The intuition for the algorithm is that the global min-cut in a graph has capacity that is small relative to the capacity of the graph, and so we are unlikely to choose an edge that is in a particular global min-cut. Last time, we showed the following.

Theorem 1 *Let S^* be a given global min-cut. Then*

$$\Pr[\textit{Random Contraction contracts no edge in } S^*] \geq \frac{1}{\binom{n}{2}}.$$

We start off the lecture by observing that we can derive from this an upper bound on the number of global min-cuts.

Theorem 2 *There are at most $\binom{n}{2}$ global min cuts.*

Proof: Notice that events in which we don't contract edges in two different global min cuts are disjoint. Let $S_1^*, S_2^*, \ldots, S_p^*$ be all global min cuts. Let C_i be the event that we don't contract any edge in S_i^*. We showed in last lecture that $Pr[C_i] \geq \frac{1}{\binom{n}{2}}$. Since these events are all disjoint $\sum_{i=1}^{p} Pr[C_i] \leq 1$, so we must have $p \leq \binom{n}{2}$. □

An example of a graph with $\binom{n}{2}$ global min cuts is an n node cycle, where each edge has the same capacity, since then any pair of edges forms a global min-cut.

We now turn to the question of how to take the Random Contraction algorithm and turn it into an algorithm that returns a global min-cut with high probability.

Theorem 3 *We can implement random contraction in $O(n^2)$ time ($O(n)$ per contraction).*

Theorem 4 *We can find a global min cut with probability $\geq 1 - \frac{1}{n}$ in time $O(n^4 \log n)$.*

Proof: Suppose we repeat Random Contraction $\binom{n}{2}\log n$ times. Then using the fact that $1-x \leq e^{-x}$, we have

$$\Pr[\text{we don't find min cut}] \leq \left(1-\frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}\log n} \leq \left(e^{-\frac{1}{\binom{n}{2}}}\right)^{\binom{n}{2}\log n} = e^{-\log n} = \frac{1}{n}.$$

So $\Pr[\text{we do find min cut}] \geq 1-\frac{1}{n}$. $\square$

1.2 Recursive random contraction

The running time of this algorithm is not very competitive, but Random Contraction (RC) has some features that we like, so we'll try to exploit those to find a better algorithm. One of the drawbacks of RC is that the probability of contracting an edge in a cut is low for large graphs and higher as the graph gets smaller. One idea is to make the graph smaller by using RC, and then use some other algorithm. In this case, we'll use as our other algorithm two runs of RC. Our algorithm is thus called *Recursive Random Contraction (RRC)*. We'll do two iterations of contracting the graph down to roughly a factor of $1/\sqrt{2}$ of its previous size, then call RRC on the remaining graph. We return the smaller of the two cuts found in the two iterations.

Recursive Random Contraction (Karger, Stein '96)

> If $|V| \leq 6$, find a min cut by exhaustive enumeration
> Else
>   For $i \leftarrow 1$ to 2
>   Run RC to get H_i of $\left\lceil 1+\frac{|V|}{\sqrt{2}}\right\rceil$ vertices
> $C_i \leftarrow \text{RRC}(H_i)$
>   Return smaller of C_1, C_2

Now we can start to analyze the algorithm.

Lemma 5 *RRC runs in $O(n^2\log n)$ time.*

Proof: The running time on n vertices is given by the recurrence

$$T(n) = 2T\left(\left\lceil 1+\frac{n}{\sqrt{2}}\right\rceil\right) + O(n^2)$$

Solving this recurrence we get

$$T(n) = O(n^2\log n).$$

$\square$

Lemma 6 *RRC finds a global min cut with probability $\Omega\left(\frac{1}{\log n}\right)$.*

Corollary 7 *Running RRC $O(\log^2 n)$ times we find a min cut with probability $\geq 1 - \frac{1}{n}$.*

To prove Lemma 6, we'll need the following

Lemma 8 *For a given global min-cut S^*,*

$$\Pr[\textit{no edge in } \delta(S^*) \textit{ is contracted when graph is contracted from } n \textit{ to } t \textit{ vertices}] \geq \frac{\binom{t}{2}}{\binom{n}{2}}.$$

Proof: We recall from last time that the probability that no edge in $\delta(S^*)$ is contracted in the ith iteration given that no edge was contracted in previous iterations is at least $1 - 2/(n - i + 1)$. Thus the desired probability is at least

$$\begin{aligned} \prod_{i=1}^{n-t}\left(1 - \frac{2}{n-i+1}\right) &= \prod_{i=1}^{n-t}\left(\frac{n-i-1}{n-i+1}\right) \\ &= \prod_{j=n}^{t+1}\left(\frac{j-2}{j}\right) \\ &= \frac{\binom{t}{2}}{\binom{n}{2}}. \end{aligned}$$

□

Now we can prove Lemma 6.

Proof of Lemma 6: Given a graph of size t, the probability that no edge in $\delta(S^*)$ is contracted when reduced to $\lceil 1 + \frac{t}{\sqrt{2}} \rceil$ vertices is at least $\frac{\lceil 1+\frac{t}{\sqrt{2}} \rceil \left(\lceil 1+\frac{t}{\sqrt{2}} \rceil - 1\right)}{t(t-1)} \geq \frac{1}{2}$.

Let $P(t)$ be the probability that RRC doesn't contract any edge in $\delta(S^*)$ in a graph of t vertices. Then

$$\begin{aligned} P(t) &\geq 1 - Pr[\text{some edge in } \delta(S^*) \text{ is contracted in either iteration 1 or 2}] \\ &\geq 1 - \left(1 - \Pr[\text{no edges are contracted when going to } \tfrac{t}{\sqrt{2}} \text{ vertices}] P\left(\left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil\right)\right)^2 \\ &\geq 1 - \left(1 - \frac{1}{2} P\left(\left\lceil 1 + \frac{t}{\sqrt{2}} \right\rceil\right)\right)^2. \end{aligned}$$

We know that $P(t) = 1$ for $t \leq 6$. Skipping the algebra needed to work out this recursion, we get

$$P(t) = \theta\left(\frac{1}{\log t}\right).$$

□

So far we've seen the following global min0cut algorithms:

- MA orderings: $O(n(m + n \log n))$
- RRC: $O(n^2 \log^2 n)$
- Hao-Orlin: $O(mn \log n)$

There is still another global min-cut algorithm that we haven't covered:

- Karger: $O(m \log^c n) = \widetilde{O}(m)$

Contractions are not easy to implement, so though they look like the simplest algorithms, they are not the clear winner. There is a comparative study of all these four algorithms by Chekuri, Goldberg, Karger, Levine and Stein (1996). The two clear winners in their study are MA orderings and Hao-Orlin, and they say that Hao-Orlin is the most robust of their codes; this is probably in part because there are a lot of good implementations of push/relabel around. This shows that a better theoretical running time doesn't necessarily imply that an algorithm is better in practice.

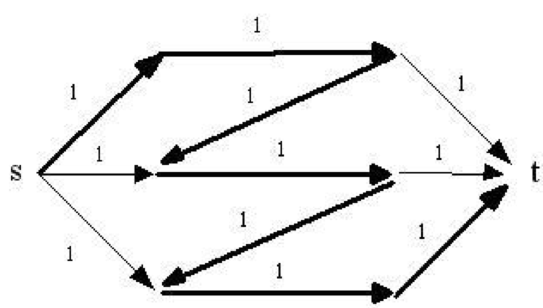
2 Efficient algorithms for max flow

2.1 Blocking flow

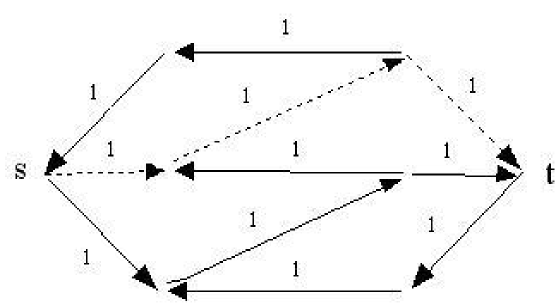
We'll work now toward one more algorithm for computing a maximum flow. It is based on the concept of a *blocking flow*.

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

Every maximum flow is obviously also a blocking flow. Is every blocking flow a maximum flow? No. The following example illustrates this:



Blocking flow



The residual graph has an augmenting path, so f is not max flow.

However, blocking flows are still useful in order to compute maximum flows. We give an algorithm below.

Let d_i = distance from a node i to the sink t.

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

Note that an arc being admissible means it is on a shortest path to t. Next time we'll discuss the following algorithm.

Dinic's algorithm (1970)

$f \leftarrow 0$
While $\exists s - t$ path in G_f
Compute distance d_i from i to t in G_f
Find blocking flow $\widetilde{f}$ in graph G_f with only admissible arcs, capacity u_{ij}^f
$f \leftarrow f + \widetilde{f}$