COMPSCI 590.1: Graph Algorithms January 27, 2015

Lecture 6

Lecturer: Debmalya Panigrahi *Scribe: Yuhao Hu*

1 Overview

In this lecture, we start introducing the randomization of (undirected) graphs. Given a graph $G = (V, E)$, one can consider the following two operations:

- Edge contraction: randomly pick an edge, then contract it.
- Edge sampling : pick every edge $e \in E$ independently with probability p_e.

The two main results studied in this lecture are the *contraction algorithm* and the *uniform sampling theorem*. Both are credited to David Karger. Through out this notes, G will always used to denote an undirected graph. In addition, we reserve n, m to represent the number of vertices and edges of G, respectively.

2 Karger's Contraction Algorithm

In this section, we first consider a baby version of Karger's contraction algorithm. Then we generalize the algorithm by allowing more flexibility in the terminating conditions. In both settings, we give tight bounds for the number of global min-cuts in graph G.

2.1 Contraction Algorithm Terminating at 2 Vertices

Definition 1. *A global min-cut of a graph G is defined as the minimum among all cuts of G.*

Lemma 1. *Fix a global min-cut $\mathscr{C}$ of G. The contraction algorithm below outputs $\mathscr{C}$ with probability $\binom{n}{2}^{-1}$.*

Contraction Algorithm
repeat {
randomly contract an edge chosen uniformly at random;
remove self loops;
} until # vertices = 2;
output the cut between the two vertices.

Proof. Let λ be the size of the global min-cut $\mathscr{C}$ and G_k be the graph after the k^{th} contraction of edges, in particular, $G_0 = G$.

Suppose that all edges in $\mathscr{C}$ are preserved and $\mathscr{C}$ is a global min-cut in G_{k-1}. In this case, let p_k be the probability of *not picking* any edge in $\mathscr{C}$ during the $(k+1)^{th}$ contraction. Then we have estimate

$$p_k = 1 - \frac{\lambda}{m_k}, \tag{1}$$

where m_k is the number of edges left in G right before the k^{th} contraction.

On the other hand, it is not hard to show that if none of the edges in $\mathscr{C}$ are picked during the k^{th} contraction, then $\mathscr{C}$ remains a global min cut in G_k. Hence, by minimality of λ, we obtain

$$d_v \geq \lambda \quad \Rightarrow \quad m_k = \frac{1}{2}\sum_{v\in V_k} d_v \geq \frac{(n-k)\lambda}{2}. \tag{2}$$

Now let P be the probability that none of the edges in $\mathscr{C}$ is contracted after the above algorithm. We have

$$p_k = 1 - \frac{\lambda}{m_k} \geq 1 - \frac{2\lambda}{(n-k)\lambda} = 1 - \frac{2}{n-k}, \tag{3}$$

$$P = \prod_{k=0}^{n-3} p_k \geq \prod_{k=0}^{n-3}(1 - \frac{2}{n-k}) = \frac{n-3}{n}\cdot\frac{n-3}{n-1}\cdots\frac{1}{3} = \binom{n}{2}^{-1}. \tag{4}$$

This completes the proof. □

Remark 1. *As a consequence of Lemma 1, one could run the contraction algorithm sufficiently many (e.g. $cn^2\log n$) times to get an overall high probability of outputting the min-cut $\mathscr{C}$. However, total running time for this is roughly at least $O(n^3)$. A natural question to ask is whether one could obtain shorter running time. In fact, it has been shown that the running time can be improved to $O(n^2)$ even $O(m)$, but we will not get into the details here.*

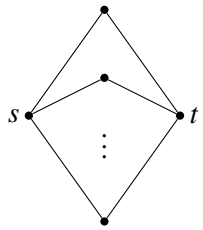
Corollary 2. *The total number of global min-cuts is $O(n^2)$.*

Proof. Note that the events of outputting each min-cut after the algorithm described in Lemma 1 are disjoint. Supposing that the total number of min-cuts is N, we must have

$$1 \geq \sum_{\mathscr{C}\text{ min-cut}} \mathbb{P}(\text{algorithm in Lemma 1 outputs } \mathscr{C}) \geq N\binom{n}{2}^{-1}. \tag{5}$$

Hence $N = O(n^2)$. □

Remark 2. *In corollary 2, we have put two restrictions on the cuts that are under counting. One is that the cuts are global, the other is that the cuts are minimal. Removing either restriction would lead to completely different bounds. For example, the number of global cuts in a graph is bounded above by 2^n by considering all possible 2-partitions of the vertices. Another example is illustrated in the figure on the right, showing that one could have 2^{n-2} $(s,t)-$min-cuts, for fixed (s,t).*



Remark 3. *The bound in corollary 2 is tight. To see this, simply consider an undirected cycle.*

2.2 Contraction Algorithm Terminating at 2α Vertices

In Lemma 1, we studied the contraction algorithm which terminates exactly when there are two vertices remaining in the graph. In fact, one can consider a broader class of algorithms by choosing the terminating conditions to be: there are 2α vertices remaining in the graph. Observe that, in doing this, our original way of output no longer works. To fix that, simply let the output be a random cut that has survived the contractions. For each α, let us call such an algorithm the $2\alpha-$*contraction algorithm*, described as follows.

$2\alpha-$**Contraction Algorithm**
repeat {
randomly contract an edge chosen uniformly at random;
remove self loops;
} until # vertices $= 2\alpha$;
output a random cut in the final graph.

Definition 2. *Given a graph G, an $\alpha-$min-cut is a cut with no more than $\alpha\lambda$ edges, where λ is the size of the global min-cut of G.*

Lemma 3. *Fix any $\alpha-$min-cut $\mathscr{C}$. The $2\alpha-$contraction algorithm outputs $\mathscr{C}$ with probability at least $n^{-O(\alpha)}$.*

Proof. The proof is similar as that of Lemma 1. Here, let p_k, m_k and P have the same meaning as before. Then we have estimate

$$p_k \geq 1 - \frac{\alpha\lambda}{m_k} \geq 1 - \frac{2\alpha\lambda}{(n-k)\lambda} = 1 - \frac{2\alpha}{n-k}, \tag{6}$$

$$P \geq 2^{-(2\alpha-1)} \cdot \prod_{k=0}^{n-2\alpha-1} p_k = 2^{-(2\alpha-1)} \cdot \frac{n-2\alpha}{n} \cdot \frac{n-2\alpha-1}{n-1} \cdots \frac{1}{2\alpha+1} \tag{7}$$

$$= 2^{-(2\alpha-1)} \cdot \frac{(2\alpha)!}{n(n-1)\cdots(n-2\alpha+1)} \geq n^{-2\alpha}. \tag{8}$$

This completes the proof. □

Corollary 4. *In a graph G, the total number of global $\alpha-$min-cuts is $n^{O(\alpha)}$.*

Proof. The proof is identical as that of Lemma 2. □

3 Karger's Uniform Sampling Theorem

In many situations we want to sparsify a graph. One obvious approach is by sampling edges. In this section, we study randomized sampling algorithms which preserve the value of every cut in the graph with high probability.

Starting with one simplest setting, let us assume that in a graph G, each edge is sampled with some probability p. If we stop here, then the expected value of each cut is only p times the true value. This problem can be easily solved by replacing each sampled edge with $1/p$ parallel edges. In doing this, the expected value of each cut in the sampled graph is exactly the true value of the cut before sampling. This can be summarized as the algorithm below.

Algorithm
sample every edge independently with probability p;
if edge e is sampled, then replace e by $1/p$ parallel edges in the sample.

Note that for any min-cut in G, the probability that this cut is preserved after sampling is roughly $1 - \frac{1}{n^2}$. However, if we take the number of cuts into consideration, the failure probability of sampling could still be high. Clearly now we want the sampling algorithm to achieve two properties:

- cut values are preserved with high probability ;
- the union bound for the probability of error is small.

For instance, it would be desirable if we could have

$$\mathbb{P}[\tilde{G}(S,\bar{S}) \notin (1\pm\varepsilon)G(S,\bar{S})] \leq \frac{1}{n^3}, \tag{9}$$

where $\tilde{G}(S,\bar{S})$ stands for the output cut and the whole LHS represents the probability of the failure of the output cut to be within ε-error of the original cut. First consider a min-cut of λ edges, say, $e_1,...e_\lambda$, each sampled with probability p. Then the general bound give

$$\mathbb{P}[\text{ sampled edges } \notin (1\pm\varepsilon)\lambda p] \leq e^{-C\cdot\varepsilon^2\cdot\mu} = e^{-C\cdot\varepsilon^2\cdot\lambda p}. \tag{10}$$

This means that setting p to be $\frac{D\log n}{\lambda\varepsilon^2}$ for some large constant D will give us high probability of preserving each global min-cut. However, question remains that how well the other cuts are preserved.

Karger's Sampling Algorithm
sample every edge independently with probability $p = \frac{D\log n}{\varepsilon^2\lambda}$;
add $1/p$ copies of every sampled edge to the output graph $\tilde{G}$.

In fact, similar to what can be obtained for global min-cuts, one can derive failure probability bounds for each $\alpha-$cut

$$\mathbb{P}[\tilde{G}(S,\bar{S}) \notin (1\pm\varepsilon)G(S,\bar{S})] \leq e^{-C\cdot\varepsilon^2\cdot\mu} = e^{-C\cdot\varepsilon^2\cdot\alpha\lambda p} = n^{-\Omega(\alpha)}. \tag{11}$$

Moreover, the total probability of failure is bounded (by choosing appropriate parameters in p):

$$\int_1^\infty n^{-\Omega(\alpha)}\cdot n^{O(\alpha)}d\alpha = \int_1^\infty n^{-\Omega(\alpha)} = n^{-\Omega(1)}. \tag{12}$$

Recall that our reason for sampling edges in a graph is that it may sparsify the graph thus give faster graph algorithms. However, the algorithm above may not perform well if λ is small. In that case, we are sampling every edge with same high probability, which is not desirable. One simple example for this is a graph which consists of two complete subgraphs connected by only one edge. In this case, the Karger's sampling algorithm simply samples every edge in the graph with probability 1. A problem with this algorithm is now clear: we are sampling every edge with same probability, regardless of the combinatorial properties of the edge. We are then motivated to introduce connectivity parameters for edges, which is the topic of the next lecture.

4 Summary

In this lecture we introduced the contraction algorithm and the uniform sampling theorem. Using the contraction algorithm we derived a tight upper bound for the number of $\alpha-$min-cuts in a graph. On the other hand, the uniform sampling algorithm outputs all the cuts in a graph with high probability. However, problem remains that the sampling algorithm may not actually sparsify a graph. Connectivity parameters will be introduced in the next lecture to solve this problem.