COMPSCI 638: Graph Algorithms September 20, 2019

Lecture 8

Lecturer: Debmalya Panigrahi *Scribe: Kevin Sun*

1 Overview

In this lecture, we introduce the global minimum cut problem and present two randomized algorithms. We also prove some structural implications of these algorithms that bound the number of minimum cuts in an undirected graph.

2 Global Minimum Cuts

In this problem, we are given an (undirected) graph $G = (V, E)$ with n vertices, m edges, and edge capacities $u(e)$; for now, we assume every edge has unit capacity. The goal is to find a cut with the least capacity, i.e., a subset of vertices with the fewest number of crossing edges. Throughout this section, we let C denote the edge set of a minimum cut, and $\lambda = |C|$ denote the size of the minimum cut.

2.1 The Contraction Algorithm

The first algorithm we present is due to Karger [Kar93]. Consider what happens when we *contract* an edge $\{u, v\}$, that is, we identify u and v as a single vertex incident to the original neighbors of u and v. If $\{u, v\}$ is not in C, then the minimum cut has been preserved in the new graph. Intuitively, the number of edges in a minimum cut is small, so it suffices to repeatedly contract edges chosen uniformly at random.

Algorithm 1 Contraction Algorithm (Karger [Kar93])

```
1: while the graph has more than two vertices do
2:     Contract an edge chosen uniformly at random.
3: return the cut specified by the two remaining vertices
```

Lemma 1. *Algorithm 1 returns a minimum cut with probability* $1/\binom{n}{2}$.

Proof. Recall that $\lambda = |C|$ denotes the number of edges in the minimum cut C. The algorithm successfully returns C if we never pick an edge in C. Let p_k denote the probability that the algorithm does not pick an edge in C when the graph has k vertices, so we have

$$p_n = \left(1 - \frac{\lambda}{m}\right) p_{n-1}.$$

Furthermore, degree of every vertex is at least λ, and the sum of degrees is exactly $2m$; this implies $m \geq n\lambda/2$. Substituting this into the above yields

$$p_n \geq \left(1 - \frac{2}{n}\right) p_{n-1}. \tag{1}$$

We solve this recurrence by noticing that if $f_k = 1/\binom{k}{2}$, then

$$f_n = \frac{1}{\binom{n}{2}} = \frac{2}{n(n-1)} = \frac{n-2}{n} \cdot \frac{2}{(n-1)(n-2)} = \left(1 - \frac{2}{n}\right) f_{n-1}.$$

So together with (1), we have $p_n \geq f_n = 1/\binom{n}{2}$. □

To boost the probability of success, we can simply run Algorithm 1 multiple times.

Theorem 2. *With high probability, the smallest cut found by running Algorithm 1 independently $O(n^2 \log n)$ times is a minimum cut.*

Proof. By Lemma 1 and the inequality $1 - x \leq e^{-x}$, the probability that this procedure fails (i.e., every run of Algorithm 1 fails to return a minimum cut) is at most

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2}\log n} \leq e^{-\log n} \approx \frac{1}{n^d}$$

where d can be any constant (at it is determined by our choice of the base of the logarithm). □

2.2 Recursive Contraction

A straightforward implementation of Algorithm 1 runs in $O(n^2)$ time, so Theorem 2 describes an algorithm that runs in $O(n^4 \log n)$ time. In this section, we will show how to improve this to nearly (up to polylogarithmic factors) $O(n^2)$.

To do this, let us observe the following: since $m \geq n\lambda/2$, the chance that a single run fails in the first step is $\lambda/m \leq 2/n$, but this probability increases over time to $2/3$. Thus, it seems unnecessary to repeat the early steps of the algorithm in every iteration. To formalize this, let us look at an algorithm due to Karger and Stein [KS96].

Algorithm 2 Recursive Contraction (Karger and Stein [KS96])

```
1: procedure RC($G = (V, E)$)
2:     if $|V| \leq 6$ then
3:         Return the global minimum cut of $G$.
4:     Contract $G$ until $|V|$ has decreased by a factor of $\sqrt{2}$.
5:     for $i = 1, 2$ do
6:         $C_i \leftarrow$ RC($G$)
7:     return the smaller of $C_1, C_2$
```

Lemma 3. *Algorithm 2 returns a minimum cut with probability $\Omega(1/\log n)$.*

Proof. We visualize the computation Algorithm 2 as a tree: at the top level, we start with the initial graph with n vertices and contract until we have $n/\sqrt{2}$ vertices, at which point we split into two branches. Each branch computes independently until $n/2$ vertices remain in each, at which point they each split for a total of four branches, and so on.

If we ever contract an edge in the minimum cut C in some branch, then that branch has failed. Let d denote the initial height and $d - i$ denote the height of the i-th branching step. Consider our initial contraction from level d to $d - 1$: the probability this succeeds is

$$\left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\cdots\left(1 - \frac{2}{n/\sqrt{2}}\right) \approx \frac{1}{2}.$$

After branching at level $d - 1$, the algorithm fails only if *both* copies fail. Thus, if p_i denotes the probability of success starting at level i, then the overall probability of failing is

$$1 - p_d \leq \frac{1}{2} + \frac{1}{2}\left(1 - p_{d-1}\right)^2.$$

Rearranging this gives us

$$p_d \geq \frac{1}{2} - \frac{1}{2}(1 - p_{n-1})^2 = \frac{1}{2}(2p_{n-1} - p_{n-1}^2) = p_{n-1} - \frac{p_{n-1}^2}{2}.$$

It is straightforward to verify $f(k) = 1/k$ satisfies

$$f(d) \leq f(d-1) - \frac{f^2(d-1)}{2},$$

so $p_d \geq f(d) = 1/d$. Since the depth of the computation tree is $d = \log_{\sqrt{2}} n$, we can conclude that the overall probability of success is $p_d = \Omega(1/\log n)$. $\square$

Lemma 4. *The running time of Algorithm 2 is $O(n^2 \log n)$.*

Proof. Contracting a graph from n vertices to $n/\sqrt{2}$ vertices requires $O(n^2)$ time, so the running time $T(n)$ of Algorithm 2 on a graph with n vertices satisfies

$$T(n) = O(n^2) + 2 \cdot T\left(\frac{n}{\sqrt{2}}\right).$$

This recurrence is solved by $T(n) = O(n^2 \log n)$. $\square$

As with Algorithm 1, we can boost the probability of success by running Algorithm 2 multiple times. This gives us the following theorem, whose proof we omit because it is essentially identical to the proof of Theorem 2.

Theorem 5. *With high probability, the smallest cut found by running Algorithm 1 independently $O(\log^2 n)$ times is a minimum cut.*

Finally, we note that both of these algorithms can be appropriately modified for edge-weighted graphs; for example, we can replace an edge with integer weight k by k parallel edges.

3 Counting Minimum Cuts

Despite its simplicity, Algorithm 1 has major consequences that have been used in a wide variety of graph algorithms. The most immediate consequence allows us to bound the number of minimum cuts in any undirected graph.

Lemma 6. *The number of minimum cuts in an undirected graph is at most $\binom{n}{2}$, and this bound is tight.*

Proof. Observe that our analysis of the contraction algorithm works for any fixed minimum cut, and only one such cut can be output by the algorithm. Thus, if $C_1, \ldots, C_t$ denotes the set of minimum cuts for some t, then by Lemma 1, we have

$$1 \geq \Pr(\text{Algorithm 1 succeeds}) = \sum_{i=1}^{t} \Pr(\text{Algorithm 1 returns } C_i) \geq \frac{t}{\binom{n}{2}}.$$

The cycle graph on n vertices is a tight example: any pair of edges forms a minimum cut. □

We can extend this to a more general bound on the number of cuts that are "close" to being minimum. Before doing so, we formalize this notion with the following definition.

Definition 1. *In a graph with minimum cut λ, an α-*minimum cut *is a cut with at most $\alpha\lambda$ edges.*

Lemma 7. *The number of α-minimum cuts in an undirected graph is at most $\binom{n}{2\alpha} = n^{O(\alpha)}$.*

Proof. Consider running the contraction algorithm and instead of stopping at two vertices, we stop at 2α vertices for some α, and then output a random cut (defined by a subset of remaining vertices). Then by the same analysis of Lemma 1, the probability we return an α-minimum cut is at least

$$\left(1 - \frac{2\alpha}{n}\right)\left(1 - \frac{2\alpha}{n-1}\right) \cdots \left(1 - \frac{2\alpha}{2\alpha+1}\right) \cdot \frac{1}{2^{2\alpha-1}}.$$

where last step comes from our final random step. One can show that this quantity is at least $1/\binom{n}{2\alpha}$, and this proves the lemma. (Note that the cycle is still a tight example.) □

4 Summary

In this lecture, we studied the contraction algorithm for global minimum cuts, as well as the improved version known as recursive contraction. We also showed how the contraction algorithm allows us to bound the number of minimum cuts in an undirected graph.

References

[Kar93] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-out algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[KS96] David R Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.