CS 6550: Randomized algorithms **Spring 2019**

Lecture 1: Karger's min-cut algorithm and the Karger-Stein algorithm

January 8, 2019

Lecturer: Eric Vigoda *Scribes: Yingjie Qian, John Lambert*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

1.1 Introduction

Definition 1.1 *Given an undirected graph $\mathcal{G} = (V, E)$, let $S \subseteq V$ and $\bar{S} = V \setminus S$. We define the cut$(S, \bar{S})$ to be the set of all edges with one end in S and the other in $\bar{S}$. We use notation $\delta(S) = \{(v, w) \in E : \forall v \in S, w \in \bar{S}\}$ for simplicity.*

In this lecture, we want to solve the **min-cut problem**: Given $G = (V, E)$, find $S \subset V$ such that $|\delta(S)|$ is minimum.

One easy way is to use max-flow. Recall the **min-s,t-cut problem**, which is given G and two vertices $s, t \in V$, find cut$(S, \bar{S})$ where $s \in S, t \in \bar{S}$, with $|\delta(S)|$ minimum. To solve *min-cut* from *min-s,t-cut*: fix $s \in V$ and run it for all $n - 1$ other vertices as t. To solve *min-cut*: we can reduce it to **max-s,t-flow** and use Edmond's Karp algorithm to solve it. We refer the readers to CS 6505 notes for details.

In this lecture, we present Karger's min-cut algorithm [1] in section 1.2.1 and the Karger-Stein algorithm [2] in section 1.3.1. We will then analyze their running times and success probabilities in section 1.2.2 and 1.3.2, respectively.

1.2 Karger's min-cut algorithm

1.2.1 The algorithm

In this section, we first introduce a new action on graph called contraction. Then, we provide some intuition before formally presenting Karger's min-cut algorithm.

Given a graph $G = (V, E)$, for an edge $e = (v, w) \in E$, we call the following action as *contracting* e to obtain a new graph, denoted by G/e:

1. Replace vertices v and w by new vertex y.
2. For all edges $(v, z) \in E$ in original graph G, replace by (y, z); for all edges $(w, z) \in E$, replace by (y, z).
3. Drop all edges $(v, w) \in E$.

Note that G/e is a multi-graph without self-loops. We have the following observations on edge contractions: For a given graph $G = (V, E)$ and $e = (v, w) \in E$, let $G' = G/e$. Consider cuts in G and cuts in G', then every cut in G' corresponds to a cut in G. For a subset $S \subset V$, where $e \in S$, then $\delta_G(S) = \delta_{G'}(S)$. Now, consider contracting a sequence of edges $e_1, e_2, \ldots, e_{n-2}$. Since the number of vertices decreases by one via each contraction, there are two big meta-vertices left. Note that the (multi-)edges between those two vertices represent the cut$(S', \bar{S}')$ in G, where the vertices in S' and $\bar{S}'$ are contracted into those two big meta-vertices respectively.

Fix a cut $(S^\star, \bar{S}^\star)$ of minimum size $|\delta(S^\star)| = k$. We want all of these edges $e_1, e_2, \ldots, e_{n-2} \notin \delta(S^\star)$. In a randomized algorithm, just take guesses at choosing edges. The minimum cut is small, so if you choose a random edge from the whole graph, it is unlikely that it will be in the cut. Below is the formal algorithm.

Algorithm 1: Karger's min-cut algorithm

```
input : Graph G = (V, E).
n ← |V(G)|, C ← E;
for i ← 1 to l·C(n,2) do
    G' ← G;
    repeat
        Choose an edge e ∈ E(G');
        G' ← G'/e;
    until |V(G')| = 2;
    if |C| ≥ |E(G')| then
        C ← E(G');
output: C.
```

1.2.2 Analysis

We can use adjacency lists or an adjacency matrix to represent the graph. Then, a single edge contraction operation can be implemented with a linear number of updates to the data structure. It follows that the running time for contracting any given graph to two vertices is $O(n^2)$ as we contract $n-2$ edges. So the total running time for the algorithm is $O(n^4)$.

Now, we will show that the success probability of the algorithm is:

$$P^* := \Pr\big(\text{algorithm outputs the global min cut}\big) \geq 1 - e^{-l}.$$

Fix $S^* \subset V$ such that cut $(S^*, \bar{S}^*)$ is of minimum size k. It suffices to show that for a single run,

$$P := \Pr\big(\text{algorithm outputs min cut } (S^*, \bar{S}^*) \text{ for a single run}\big) \geq \frac{1}{\binom{n}{2}}.$$

Suppose we can prove the above inequality, since we run the algorithm $l\binom{n}{2}$ times and output the best cut found, the probability of all $l\binom{n}{2}$ runs do not find the min cut is $(1-P)^{l\binom{n}{2}}$. Thus,

$$P^* \geq 1 - (1-P)^{l\binom{n}{2}} \geq 1 - e^{-Pl\binom{n}{2}} \geq 1 - e^{-l}.$$

We are left to show $P \geq \frac{1}{\binom{n}{2}}$. By observation, $P = \Pr\big(e_1, e_2, \ldots, e_{n-2} \notin \delta(S^\star)\big)$.

Then,

$$\Pr(e_1 \notin \delta(S^*)) = 1 - \frac{k}{|E(G)|}.$$

We now claim that G has minimum degree k, otherwise G has cut of size smaller than k, which is the set of all edges that has one fixed minimum degree vertex as one end. Then, $|E(G)| = \frac{1}{2} \sum_{v \in V(G))} \deg(v) \geq \frac{kn}{2}$. Thus,

$$\Pr(e_1 \notin \delta(S^*)) = 1 - \frac{k}{|E(G)|} \geq 1 - \frac{k}{kn/2} = 1 - \frac{2}{n}.$$

G/e_1 still has minimum degree k, otherwise G/e_1 has cut of size smaller than k, which corresponds to cut of size smaller than k in G. Then, $|V(G/e_1)| = n-1$ and $|E(G/e_1)| \geq \frac{k(n-1)}{2}$. Thus,

$$\Pr\big(e_2 \notin \delta(S^\star) \mid e_1 \notin \delta(S^\star)\big) \geq 1 - \frac{k}{k(n-1)/2} = 1 - \frac{2}{n-1}.$$

Similarly, for $i = 3, 4, \ldots, n-2$,

$$\Pr\big(e_i \notin \delta(S^\star) \mid e_j \notin \delta(S^\star), \forall j < i\big) \geq 1 - \frac{k}{k(n-(i-1))/2} = 1 - \frac{2}{n-i+1}.$$

Thus,

$$\begin{aligned}
P &= \Pr\big(e_1, e_2, \ldots, e_{n-2} \notin \delta(S^\star)\big) \\
&= Pr(e_1 \notin \delta(S^\star)) \times Pr(e_2 \notin \delta(S^\star) \mid e_1 \notin \delta(S^\star)) \times Pr(e_3 \notin \delta(S^\star) \mid e_1, e_2 \notin \delta(S^\star)) \times \ldots \\
&\geq (1 - \frac{2}{n}) \times (1 - \frac{2}{n-1}) \times (1 - \frac{2}{n-2}) \times \cdots \times (1 - \frac{2}{3}) \\
&= (\frac{n-2}{n}) \times (\frac{n-3}{n-1}) \times (\frac{n-4}{n-2}) \times \cdots \times (\frac{2}{4}) \times (\frac{1}{3}) \\
&= \frac{2}{n \times (n-1)}. \\
&= \frac{1}{\binom{n}{2}} \qquad \text{as desired.}
\end{aligned}$$

Note that the algorithm also implies that for any graph G, there are at most $\binom{n}{2}$ minimum cuts.

1.3 The Karger-Stein algorithm

Consider the calculations in the last section. During the sequence of edge contractions, the probability of choosing right edges is high at the beginning but low later since there are small number of vertices left. The Karger-Stein algorithm uses the general framework of that in the Karger's mincut algorithm but runs the "later" part more times.

1.3.1 The algorithm

Suppose we contract edges from n vertices to l vertices twice. We can choose l to have the probability of success is about $\frac{1}{2}$. So in expectation, one of the two runs work. Then we recurse each of these bifurcates. Take best of two, then pop back up from the recursion.

Now, we want to choose l. Similar to the previous calculations, the probability of success of running from n vertices to l is

$$\frac{n-2}{n} \times \frac{n-3}{n-1} \times \cdots \times \frac{l-1}{l+1} = \frac{l(l-1)}{n(n-1)} = \frac{\binom{l}{2}}{\binom{n}{2}}.$$

Then, we have $\frac{\binom{l}{2}}{\binom{n}{2}} \geq \frac{1}{2}$ by taking $l = \frac{n}{\sqrt{2}} + 1$. Below is the algorithm:

Algorithm 2: The Karger-Stein algorithm: KargerStein(G)

input : Graph G with n vertices.
if $n \geq 6$ **then**
  Do 2 runs and **output** the best of the 2:
  Use Karger's mincut algorithm to contract edges until $\frac{n}{\sqrt{2}} + 1$ vertices are left, denoted as G';
  Recurse KargerStein(G').

Note that 6 is chosen in the algorithm as $\frac{6}{\sqrt{2}} + 1 \geq 2$.

1.3.2 Analysis

First, let us calculate the new running time for running it once. Then,

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

By the Master Theorem, $T(n) = O(n^2 \log n)$.

Now, we can also write a recurrence relation for the success probability. The probability of success to contract from n vertices to $\frac{n}{\sqrt{2}} + 1$ is no smaller than a half. Thus,

$$P(n) \geq 1 - \big(1 - \frac{1}{2}P(\frac{n}{\sqrt{2}})\big)^2.$$

One can easily use induction to verify that $P(n) = \Omega(\frac{1}{\ln n})$. So if we do $O(\log^2 n)$ runs, the success probability is at least $1 - \frac{1}{\text{poly}(n)}$. Hence, the total running time of the algorithm is $O(n^2 \log^3 n)$.

References

[1] Karger, David R.. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1993.

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