**CSE 632: Analysis of Algorithms II: Randomized Algorithms** **Spring 2024**

# Lecture 4: Global Minimum Cut

*Lecturer: Zongchen Chen*

## 1 Global Min Cut Problem

**Definition 1.** Let $G = (V, E)$ be a graph. A *cut* $(S, V \setminus S)$ is a bipartition of $V$ where $S \subseteq V$. The *cut-set* of a cut $(S, V \setminus S)$ is defined as

$$\delta(S) = E(S, V \setminus S) = \{uv \in E : u \in S, v \in V \setminus S\}.$$

Note that $\delta(S) = \delta(V \setminus S)$. We consider the problem of finding a global min cut of a given graph.

*Global min cut problem*: Given a graph $G = (V, E)$, find a cut $(S, V \setminus S)$ where $S \neq \emptyset, V$ which minimizes $|\delta(S)|$.

A closely related problem is the min $s$-$t$ cut problem.

*Min s-t cut problem*: Given a graph $G = (V, E)$ and two vertices $s, t \in V$, find a cut $(S, V \setminus S)$ where $s \in S$, $t \in V \setminus S$ which minimizes $|\delta(S)|$.

One can solve the min $s$-$t$ cut problem using the max-flow min-cut theorem and any polynomial-time algorithm for max $s$-$t$ flow; the current fastest algorithm runs in $O(m^{1+o(1)})$ time. One can solve the global min cut problem by solving $n - 1$ min $s$-$t$ cut problems: If $V = \{v_1, \dots, v_n\}$, then we solve min $v_1$-$v_i$ cut for each $i = 2, \dots, n$ and output the best cut among these $n - 1$ cuts.

## 2 Edge Contraction

In the rest of this note, we consider multigraphs without self-loops, i.e., there are possibly multiple edges between a pair of distinct vertices but no edges $vv$ for $v \in V$. For a multigraph $G = (V, E)$ without self-loops, denote the number of vertices by $n = |V|$, and the number of edges by $m = |E|$.

**Definition 2** (Edge Contraction)**.** For a multigraph $G = (V, E)$ without self-loops and an edge $e = \{u, v\} \in E$, define $G/e$ to be the graph resulted from contraction of $e$ by:

(1) Replace $u, v$ by a new vertex $w$;

(2) Replace every edge $ux$ or $vx$ where $x \in V \setminus \{u, v\}$ by a new edge $wx$.

Note that after the edge contraction, the new graph $G/e$ is a multigraph without self-loops.

**Observation 3.** Suppose $G = (V, E)$, $e = uv \in E$, and $G' = (V', E') = G/e$.

(1) If $|V| = n$, then $|V'| = n - 1$;

(2) Cuts $(S', V' \setminus S')$ in $G'$ are in one-to-one correspondence to cuts $(S, V \setminus S)$ in $G$ where $uv \notin \delta(S)$, equivalently either $\{u, v\} \subseteq S$ or $\{u, v\} \subseteq V \setminus S$. Moreover, $|\delta_{G'}(S')| = |\delta_G(S)|$ under this correspondence.

Suppose $(S^*, V \setminus S^*)$ is a global min cut of $G$ (note that there could be multiple global min cuts). Our ideas are as follows.

1. If we know in advance an edge $e \notin \delta(S^*)$ (i.e., either $\{u, v\} \subseteq S$ or $\{u, v\} \subseteq V \setminus S$), then it suffices to solve global min cut on $G/e$, which is a smaller graph.

2. We do not know if any edge $e \notin \delta(S^*)$ or not, but we know $(S^*, V \setminus S^*)$ is a global min cut, so $|\delta(S^*)|$ should be small; in particular, (we hope) a random edge $e$ would satisfy $e \notin \delta(S^*)$.

3. An algorithm can work as follows: in each step we pick a random edge and contract it, until two (super) vertices $a, b$ remain, and $(\{a\}, \{b\})$ would correspond to a cut $(S, V \setminus S)$ back in the original graph $G$ by Observation 3.

# 3 Karger's Algorithm

**Algorithm 1** Karger's min cut algorithm

**Input:** $G = (V, E)$ a multigraph without self-loops

```
1: repeat
2:     Choose an edge e ∈ E uniformly at random;
3:     G ← G/e;
4: until |V| = 2
return the cut corresponding to the final two vertices
```

**Lemma 4.** *Let $G = (V, E)$ be a multigraph without self-loops, and $(S^*, V \setminus S^*)$ be a global min cut of $G$. Then we have*

$$\Pr\left(\text{Algorithm 1 outputs } (S^*, V \setminus S^*)\right) \geq \frac{1}{\binom{n}{2}}.$$

*Proof.* Denote the sequence of edges chosen and contracted by Algorithm 1 as $e_0, e_1, \ldots, e_{n-3}$; note that we need to contract $n-2$ edges to get down to two vertices. Observe that Algorithm 1 outputs $(S^*, V \setminus S^*)$ if and only if $e_0, e_1, \ldots, e_{n-3} \notin \delta(S^*)$; this follows from Observation 3. By the chain rule, we have

$$\begin{aligned} & \Pr\left(\text{Algorithm 1 outputs } (S^*, V \setminus S^*)\right) \\ = & \Pr\left(e_0, e_1, \ldots, e_{n-3} \notin \delta(S^*)\right) \\ = & \Pr\left(e_0 \notin \delta(S^*)\right) \Pr\left(e_1 \notin \delta(S^*) \mid e_0 \notin \delta(S^*)\right) \Pr\left(e_2 \notin \delta(S^*) \mid e_0, e_1 \notin \delta(S^*)\right) \\ & \cdots \Pr\left(e_{n-3} \notin \delta(S^*) \mid e_0, e_1, \ldots, e_{n-4} \notin \delta(S^*)\right). \end{aligned} \tag{1}$$

Let's look at the first term:

$$\Pr\left(e_0 \notin \delta(S^*)\right) = 1 - \Pr\left(e_0 \in \delta(S^*)\right) = 1 - \frac{k}{m}, \tag{2}$$

where $k = |\delta(S^*)|$ and $m = |E|$.

**Fact 5.** *(1) The average degree of a graph $G = (V, E)$ is given by*

$$\bar{d} = \frac{1}{n} \sum_{v \in V} \deg(v) = \frac{2m}{n}.$$

*(2) Since $(\{v\}, V \setminus \{v\})$ is a cut of size $\deg(v)$ for all $v \in V$, it holds*

$$k = |\delta(S^*)| \leq \min_{v \in V} \deg(v) \leq \bar{d}.$$

*(3) Combining (1) and (2), we get*

$$\frac{k}{m} \leq \frac{2}{n}.$$

We deduce from Eq. (2) and Fact 5 that

$$\Pr\left(e_0 \notin \delta(S^*)\right) \geq 1 - \frac{2}{n}.$$

For the second term, suppose $e_0 \notin \delta(S^*)$ is given and let $G' = (V', E') = G/e_0$. Since $e_0 \notin \delta(S^*)$ we deduce from Observation 3 that $|\delta_{G'}((S^*)')| = |\delta_G(S^*)| = k$ and $((S^*)', V' \setminus (S^*)')$ is a min cut of $G'$. Recalling $|V'| = n-1$ and letting $m' = |E'|$, we deduce from Fact 5 that for any $e_0 \notin \delta(S^*)$,

$$\Pr\left(e_1 \notin \delta(S^*) \mid e_0\right) = 1 - \frac{k}{m'} \geq 1 - \frac{2}{n-1}.$$

In particular,

$$\Pr\left(e_1 \notin \delta(S^*) \mid e_0 \notin \delta(S^*)\right) \geq 1 - \frac{2}{n-1}.$$

More generally, for each $i = 0, 1, \ldots, n-3$ we have

$$\Pr\left(e_i \notin \delta(S^*) \mid e_0, e_1, \ldots, e_{i-1} \notin \delta(S^*)\right) \geq 1 - \frac{2}{n-i}.$$

Back to Eq. (1), we get

$$\begin{aligned}
\Pr\left(\text{Algorithm 1 outputs } (S^*, V \setminus S^*)\right) &\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\cdots\left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right) \\
&= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \cdots \cdot \frac{2}{4} \cdot \frac{1}{3} \\
&= \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}},
\end{aligned}$$

as claimed. □

**Success probability.** Lemma 4 shows that Algorithm 1 succeeds with probability at least $\Omega(n^{-2})$. We can boost the success probability by running Algorithm 1 for $\binom{n}{2} \cdot c \log n$ times where $c > 0$ is some constant, and output the best cut among them. Then we have

$$\Pr\left(\text{none of } \binom{n}{2} \cdot c \log n \text{ runs output } (S^*, V \setminus S^*)\right) \leq \left(1 - \frac{1}{\binom{n}{2}}\right)^{\binom{n}{2} \cdot c \log n} \leq e^{-c \log n} = \frac{1}{n^c}.$$

Thus, the success probability is

$$\Pr\left(\text{at least one of } \binom{n}{2} \cdot c \log n \text{ runs outputs } (S^*, V \setminus S^*)\right) \geq 1 - \frac{1}{n^c}.$$

This means that to guarantee a success probability at least $1 - 1/\mathrm{poly}(n)$, it suffices to run Algorithm 1 for $O(n^2 \log n)$ times.

**Running time.** Using, e.g., the adjacency matrix representation, every edge contraction takes $O(n)$ time. Thus, each run of Algorithm 1 takes $O(n^2)$ time. The overall running time to get success probability at least $1 - 1/\mathrm{poly}(n)$ is $O(n^4 \log n)$.

## 4 Karger–Stein Algorithm

Looking closer at Karger's algorithm (Algorithm 1), we notice that initial edge contractions are likely correct (meaning $e \notin \delta(S^*)$), for example,

$$\Pr(e_0 \notin \delta(S^*)) \geq 1 - \frac{2}{n} \quad \text{and} \quad \Pr(e_1 \notin \delta(S^*) \mid e_0 \notin \delta(S^*)) \geq 1 - \frac{2}{n-1}.$$

Meanwhile, later edge contractions are much less likely to be correct, for example,

$$\Pr(e_{n-4} \notin \delta(S^*) \mid e_0, e_1, \ldots, e_{n-5} \notin \delta(S^*)) \geq \frac{2}{4} \quad \text{and} \quad \Pr(e_{n-3} \notin \delta(S^*) \mid e_0, e_1, \ldots, e_{n-4} \notin \delta(S^*)) \geq \frac{1}{3}.$$

Thus, instead of running the whole Algorithm 1 for multiple times, a better way is to run initial edge contractions fewer times while later contractions more times. We calculate the probability that the min cut $(S^*, V \setminus S^*)$ survives after edge contractions until $\ell$ vertices remain:

$$\begin{aligned}
&\Pr((S^*, V \setminus S^*) \text{ survives down to } \ell \text{ vertices}) \\
={}& \Pr(e_0, e_1, \ldots, e_{n-\ell-1} \notin \delta(S^*)) \\
\geq{}& \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{\ell+2}\right)\left(1 - \frac{2}{\ell+1}\right) \\
={}& \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \cdots \cdot \frac{\ell}{\ell+2} \cdot \frac{\ell-1}{\ell+1} \\
={}& \frac{\ell(\ell-1)}{n(n-1)} = \frac{\binom{\ell}{2}}{\binom{n}{2}}.
\end{aligned}$$

If we choose $\ell \approx n/\sqrt{2}$, then this probability is at least $1/2$. This means, after applying random edge contractions until $n/\sqrt{2}$ vertices are left, $(S^*, V \setminus S^*)$ remains a global min cut with probability at least $1/2$.

**Algorithm 2** $\mathsf{FastMinCut}(G)$

> **for** $i = 1, 2$ independently **do**
> 
> $\quad G_i \leftarrow$ Apply random edge contractions to $G$ until $n/\sqrt{2}$ vertices remain;
> 
> **end for**
> 
> $\quad$ **return** $\min\{\mathsf{FastMinCut}(G_1), \mathsf{FastMinCut}(G_2)\}$

**Success probability.** Define $P(n)$ to be the probability of success of Algorithm 2 on *any* $n$-vertex graph. Note that Algorithm 2 fails to output a global min cut if and only if both attempts (i.e., $G_1$ and $G_2$) fail. For each $i = 1, 2$, with probability at least $1/2$ the cut $(S^*, V \setminus S^*)$ remains to be a global min on $G_i$, and with probability at least $P(n/\sqrt{2})$ the recursive call $\mathsf{FastMinCut}(G_i)$ outputs a global min cut. Hence, we have the recursion

$$1 - P(n) \leq \left(1 - \frac{1}{2}P\left(\frac{n}{\sqrt{2}}\right)\right)^2.$$

Solving the recursion gives $P(n) = \Omega(1/\log n)$.

**Running time.** The running time of Algorithm 2 satisfies the recursion

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

Therefore, $T(n) = O(n^2 \log n)$. To get success probability at least $1 - 1/\mathrm{poly}(n)$, we run Algorithm 2 for $O(\log^2 n)$ times and the overall running time is $O(n^2 \log^3 n)$.