Chapter 14

Randomized Algorithms III – Min Cut

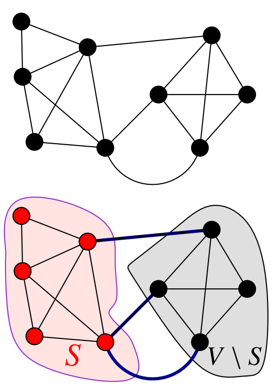
NEW CS 473: Theory II, Fall 2015
October 13, 2015

14.1 Min Cut

14.1.1 Problem Definition

14.2 Min cut

14.2.0.1 Min cut



$\mathsf{G} = (V, E)$: undirected graph, n vertices, m edges.
Interested in ***cuts*** in G.

Definition 14.2.1. ***cut*** in G: a partition of V: S and $V \setminus S$.
Edges of the cut:

$$(S, V \setminus S) = \left\{ uv \;\middle|\; u \in S, v \in V \setminus S, \text{ and } uv \in E \right\},$$

$|(S, V \setminus S)|$ is *size of the cut*

minimum cut / ***mincut***: cut in graph with min size.

14.2.0.2 Some definitions

(A) ***conditional probability*** of X given Y is $\mathbf{Pr}\left[X = x \,|Y = y\right] = \frac{\mathbf{Pr}\left[(X=x)\cap(Y=y)\right]}{\mathbf{Pr}\left[Y=y\right]}$.

$$\mathbf{Pr}\left[X = x \cap Y = y\right] = \mathbf{Pr}\left[X = x \;\middle|\; Y = y\right] \cdot \mathbf{Pr}[Y = y].$$

(B) X, Y events are ***independent***, if $\mathbf{Pr}\Big[X = x \cap Y = y\Big] = \mathbf{Pr}\Big[X = x\Big] \cdot \mathbf{Pr}\Big[Y = y\Big]$.
$\implies \mathbf{Pr}\Big[X = x \,\Big|\, Y = y\Big] = \mathbf{Pr}\Big[X = x\Big]$.

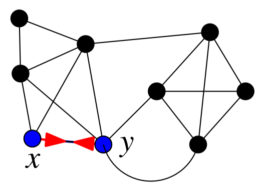
14.2.0.3 Some more probability

Lemma 14.2.2. *$\mathcal{E}_1, \ldots, \mathcal{E}_n$: n events (not necessarily independent). Then,*

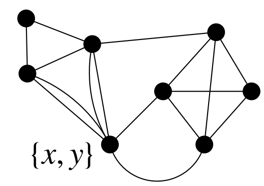
$$\mathbf{Pr}\Big[\cap_{i=1}^{n} \mathcal{E}_i\Big] = \mathbf{Pr}\Big[\mathcal{E}_1\Big] * \mathbf{Pr}\Big[\mathcal{E}_2 \,|\mathcal{E}_1\Big] * \mathbf{Pr}\Big[\mathcal{E}_3 \,\Big|\, \mathcal{E}_1 \cap \mathcal{E}_2\Big] * \ldots$$
$$* \, \mathbf{Pr}\Big[\mathcal{E}_n \,\Big|\, \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-1}\Big].$$

14.3 The Algorithm

14.3.0.1 Edge contraction...



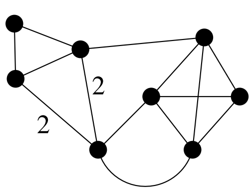
G:

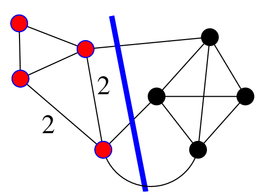


G$/xy$:

(A) ***edge contraction***: $e = xy$ in G.
(B) ... merge x, y into a single vertex.
(C) ...remove self loops.
(D) ... parallel edges – ***multi-graph***.
(E) ... weights/ multiplicities on the edges.

14.3.0.2 Min cut in weighted graph





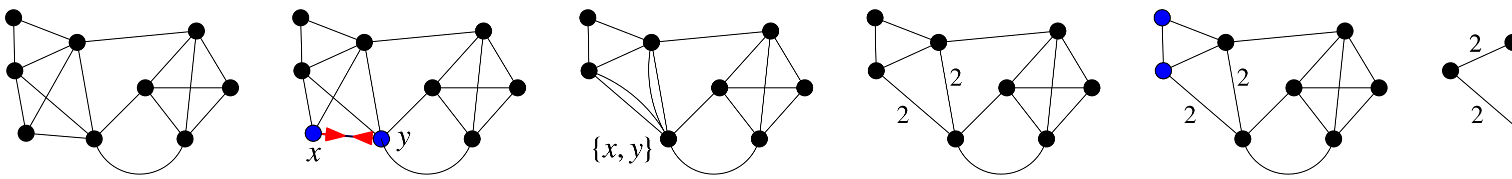
Edge contraction implemented in $O(n)$ time:
(A) Graph represented using adjacency lists.
(B) Merging the adjacency lists of the two vertices being contracted.
(C) Fix adjacency list of vertices connected to x, y.
[Use radix sort.]
(D) Include edge weight in computing cut weight.

14.3.0.3 Cuts under contractions

Observation 14.3.1. *(A) A cut in* G$/xy$ *is a valid cut in* G.
(B) There $\exists$ cuts in G *are not in* G$/xy$.
(C) The cut $S = \{x\}$ is not in G$/xy$.
(D) $\implies$ size mincut in G$/xy$ $\geq$ *mincut in* G.

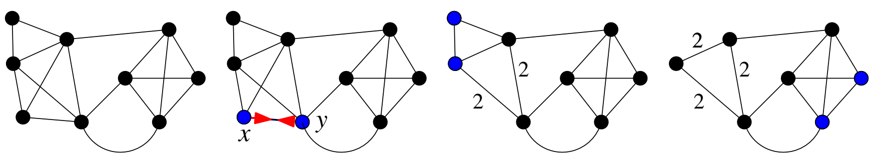
(A) **Idea**: Repeatedly perform edge contractions (benefits: shrink graph)...
(B) Every vertex in contracted graph is a connected component in the original graph.)

14.3.0.4 Contraction

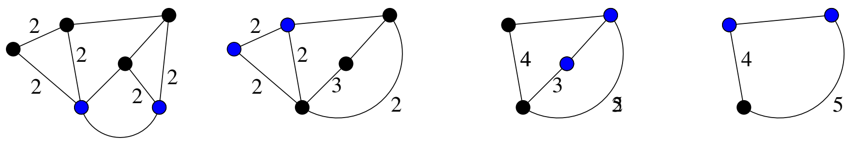


(2) (3) (4) (5) (7)

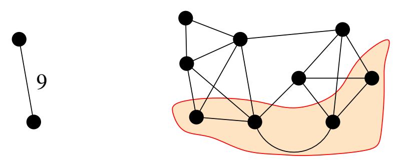
14.3.0.5 Contraction - all together now



(a) (b) (c) (d)

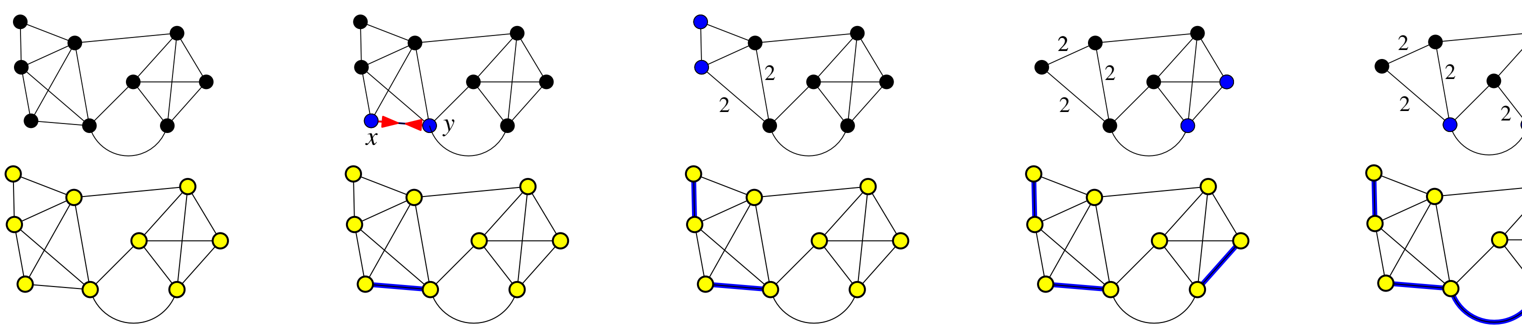


(e) (f) (g) (h)

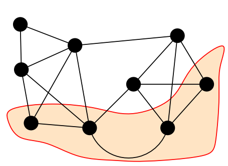


(i) (j)

14.3.0.6 Another interpretation



14.3.0.7 But...



(A) Not min cut!
(B) Contracted wrong edge somewhere...
(C) If never contract an edge in the cut...

(D) ...get min cut in the end!
(E) We might still get min cut even if we contract edge min cut. Why???

14.3.1 The resulting algorithm

14.3.1.1 The algorithm...

```
Algorithm MinCut(G)
        G_0 ← G
        i = 0
        while G_i has more than two vertices do
            e_i ← random edge from E(G_i)
            G_{i+1} ← G_i/e_i
            i ← i + 1
        Let (S, V \ S) be the cut in the original graph
                  corresponding to the single edge in G_i
        return (S, V \ S).
```

14.3.1.2 How to pick a random edge?

Lemma 14.3.2. *$X = \{x_1, \ldots, x_n\}$: elements, $\omega(x_i)$: integer positive weight.*

Pick randomly, in $O(n)$ time, an element $\in X$, with prob picking x_i being $\omega(x_i)/W$, where $W = \sum_{i=1}^{n} \omega(x_i)$.

Proof: Randomly choose $r \in [0, W]$.

Precompute $\beta_i = \sum_{k=1}^{i} \omega(x_k) = \beta_{i-1} + \omega(x_i)$.

Find first index i, $\beta_{i-1} < r \leq \beta_i$. Return x_i. ■

(A) Edges have weight...
(B) ...compute total weight of each vertex (adjacent edges).
(C) Pick randomly a vertex by weight.
(D) Pick random edge adjacent to this vertex.

14.3.2 Analysis

14.3.2.1 The probability of success

14.3.2.2 Lemma...

Lemma 14.3.3. G*: mincut of size k and n vertices, then* $|\mathsf{E}(\mathsf{G})| \geq \frac{kn}{2}$.

Proof: Each vertex degree is at least k, otherwise the vertex itself would form a minimum cut of size smaller than k. As such, there are at least $\sum_{v \in V} \text{degree(v)}/2 \geq nk/2$ edges in the graph. ■

14.3.2.3 Lemma...

Lemma 14.3.4. *If we pick in random an edge e from a graph G, then with probability at most $\frac{2}{n}$ it belong to the minimum cut.*

Proof: There are at least $nk/2$ edges in the graph and exactly k edges in the minimum cut. Thus, the probability of picking an edge from the minimum cut is smaller then $k/(nk/2) = 2/n$. ■

14.3.2.4 Lemma

Lemma 14.3.5. MinCut *outputs the mincut with prob.* $\geq \dfrac{2}{n(n-1)}$.

Proof
(A) $\mathcal{E}_i$: event that e_i is not in the minimum cut of G_i.
(B) **MinCut** outputs mincut if all the events $\mathcal{E}_0, \ldots, \mathcal{E}_{n-3}$ happen.
(C) $\mathbf{Pr}\left[\mathcal{E}_i \,\middle|\, \mathcal{E}_0 \cap \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}\right] \geq 1 - \dfrac{2}{|V(G_i)|} = 1 - \dfrac{2}{n-i}$.

$$\implies \Delta = \mathbf{Pr}[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-3}] = \mathbf{Pr}[\mathcal{E}_0] \cdot \mathbf{Pr}\left[\mathcal{E}_1 \,\middle|\, \mathcal{E}_0\right] \cdot \mathbf{Pr}\left[\mathcal{E}_2 \,\middle|\, \mathcal{E}_0 \cap \mathcal{E}_1\right] \cdot \ldots \cdot \mathbf{Pr}\left[\mathcal{E}_{n-3} \,\middle|\, \mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-4}\right]$$

14.3.2.5 Proof continued...

As such, we have

$$\begin{aligned}
\Delta &\geq \prod_{i=0}^{n-3}\left(1 - \frac{2}{n-i}\right) = \prod_{i=0}^{n-3} \frac{n-i-2}{n-i} \\
&= \frac{n-2}{n} * \frac{n-3}{n-1} * \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} \\
&= \frac{2}{n \cdot (n-1)}.
\end{aligned}$$

14.3.2.6 Some math restated...

$$\begin{aligned}
\alpha &= \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots\left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right) \\
&= \frac{n-2}{n} \cdot \frac{(n-1)-2}{n-1} \cdot \frac{(n-2)-2}{n-2} \cdots \frac{4-2}{4} \cdot \frac{3-2}{3} \\
&= \frac{n-2\cancel{n-2}}{n} \cdot \frac{n-3\cancel{n-3}}{n-1} \cdot \frac{n-4\cancel{n-4}}{n-2\cancel{n-2}} \cdot \frac{n-5\cancel{n-5}}{n-3\cancel{n-3}} \cdots \frac{3\cancel{3}}{5\cancel{5}} \cdot \frac{2}{4\cancel{4}} \cdot \frac{1}{3\cancel{3}} \\
&= \frac{}{n} \cdot \frac{}{n-1} \cdot \frac{}{} \cdot \frac{}{} \cdots \frac{}{} \cdot \frac{2}{} \cdot \frac{1}{} \\
&= \frac{2}{n(n-1)}
\end{aligned}$$

14.3.2.7 Running time analysis.
14.3.2.8 Running time analysis...

Observation 14.3.6. MinCut *runs in* $O(n^2)$ *time.*

Observation 14.3.7. *The algorithm always outputs a cut, and the cut is not smaller than the minimum cut.*

Definition 14.3.8. ***Amplification***: running an experiment again and again till the things we want to happen, with good probability, do happen.

14.3.2.9 Getting a good probability

MinCutRep: algorithm runs **MinCut** $n(n-1)$ times and return the minimum cut computed.

Lemma 14.3.9. *probability* **MinCutRep** *fails to return the minimum cut is* < 0.14.

Proof: **MinCut** fails to output the mincut in each execution is at most $1 - \frac{2}{n(n-1)}$.

MinCutRep fails, only if all $n(n-1)$ executions of **MinCut** fail.

$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)} \leq \exp\left(-\frac{2}{n(n-1)} \cdot n(n-1)\right) = \exp(-2) < 0.14$, since $1 - x \leq e^{-x}$ for $0 \leq x \leq 1$. ∎

14.3.2.10 Result

Theorem 14.3.10. *One can compute mincut in* $O(n^4)$ *time with constant probability to get a correct result. In* $O(n^4 \log n)$ *time the minimum cut is returned with high probability.*

14.4 A faster algorithm

14.4.0.1 Faster algorithm

Why **MinCutRep** needs so many executions?

Probability of failure in first ν iterations is

$$\begin{aligned}
\mathbf{Pr}\left[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{\nu-1}\right] &\geq \prod_{i=0}^{\nu-1}\left(1 - \frac{2}{n-i}\right) = \prod_{i=0}^{\nu-1} \frac{n-i-2}{n-i} \\
&= \frac{n-2}{n} * \frac{n-3}{n-1} * \frac{n-4}{n-2} \ldots \\
&= \frac{(n-\nu)(n-\nu-1)}{n \cdot (n-1)}.
\end{aligned}$$

$\implies$ $\nu = n/2$: Prob of success $\approx 1/4$.

$\implies$ $\nu = n - \sqrt{n}$: Prob of success $\approx 1/n$.

14.4.0.2 Faster algorithm...

Insight

(A) As the graph get smaller probability for bad choice increases.
(B) Currently do the amplification from the outside of the algorithm.
(C) Put amplification directly into the algorithm.

14.4.1 Contract...

14.4.1.1 Contract(G, t) shrinks G till it has only t vertices. FastCut computes the minimum cut using Contract.

```
Contract( G, t )
    while |(G)| > t do
        Pick random edge
            e in G.
        G ← G/e
    return G
```

```
FastCut(G = (V, E))
    // G -- multi-graph
    n ← |V(G)|
    if n ≤ 6 then
        return mincut G (brute force)
    t ← ⌈1 + n/√2⌉
    H_1 ← Contract(G, t)
    H_2 ← Contract(G, t)
    /* Contract is randomized */
    X_1 ← FastCut(H_1),
    X_2 ← FastCut(H_2)
    return mincut of X_1 and X_2.
```

14.4.1.2 Lemma...

Lemma 14.4.1. *The running time of* FastCut(G) *is* $O(n^2 \log n)$*, where* $n = |V(G)|$.

Proof: Well, we perform two calls to Contract(G, t) which takes $O(n^2)$ time. And then we perform two recursive calls on the resulting graphs. We have:

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

The solution to this recurrence is $O(n^2 \log n)$ as one can easily (and should) verify. ∎

14.4.1.3 Success at each step

Lemma 14.4.2. *Probability that mincut in contracted graph is original mincut is at least* $1/2$.

Proof: Plug in $\nu = n - t = n - \left\lceil 1 + n/\sqrt{2} \right\rceil$ into success probability:

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-t}\right] &\geq \frac{t(t-1)}{n \cdot (n-1)} \\ &= \frac{\left\lceil 1 + n/\sqrt{2} \right\rceil \left(\left\lceil 1 + n/\sqrt{2} \right\rceil - 1\right)}{n(n-1)} \geq \frac{1}{2}.\end{aligned}$$

14.4.1.4 Probability of success...

Lemma 14.4.3. FastCut *finds the minimum cut with probability larger than* $\Omega(1/\log n)$.

See class notes for a formal proof. We provide a more elegant direct argument shortly.

14.4.1.5 Amplification

Lemma 14.4.4. *Running* FastCut *repeatedly* $c \cdot \log^2 n$ *times, guarantee that the algorithm outputs mincut with probability* $\geq 1 - 1/n^2$.

c is a constant large enough.

Proof: (A) FastCut succeeds with prob $\geq c'/\log n$, c' is a constant.

(B) ...fails with prob. $\leq 1 - c'/\log n$.
(C) ...fails in m reps with prob. $\leq (1 - c'/\log n)^m$. But then
$$(1 - c'/\log n)^m \leq \left(e^{-c'/\log n}\right)^m \leq e^{-mc'/\log n} \leq \frac{1}{n^2},$$
for $m = \left(2\log^2 n\right)/c'$.

14.4.1.6 Theorem

Theorem 14.4.5. *One can compute the minimum cut in a graph* G *with* n *vertices in* $O(n^2 \log^3 n)$ *time. The algorithm succeeds with probability* $\geq 1 - 1/n^2$.

Proof: We do amplification on **FastCut** by running it $O(\log^2 n)$ times. The running time bound follows from lemma... ∎