

Lecture 10

Prof. Nick Harvey

University of British Columbia

In Lecture 2 we discussed the Max Cut problem, which is NP-complete, and we presented a very simple algorithm that gives a $1/2$ approximation. Today we will discuss the **Min Cut problem**, which is in P, and we will present a very simple randomized algorithm to solve it exactly.

My reasons for presenting this algorithm are:

- it illustrates that non-trivial optimization problems can sometimes be solved by very simple algorithms,
- the analysis is quite interesting, and
- most importantly, it has useful consequences that will allow us to present an even more amazing result in the next lecture.

1 Minimum Cuts

Let $G = (V, E)$ be an undirected graph. As before, for every $U \subseteq V$ we define

$$\delta(U) = \{ uv \in E : u \in U \text{ and } v \notin U \}.$$

The **Min Cut problem** is to solve

$$\min \{ |\delta(U)| : \emptyset \neq U \subsetneq V \}.$$

Here we are minimizing over all subsets U of the vertices, except for $U = \emptyset$ and $U = V$ because those two uninteresting sets have $|\delta(U)| = 0$. The Min Cut problem is equivalent to the problem

$$\min \{ |F| : F \subseteq E \text{ s.t. } G \setminus F \text{ is disconnected} \}.$$

To see the equivalence, note that any set $\delta(U)$ is a disconnecting set F , and given any disconnecting set F , we can find a cut $\delta(U) \subseteq F$ by letting U be any connected component in $G \setminus F$.

You should not confuse the Min Cut problem and the **Min s - t Cut problem**. In the latter problem, there are two distinguished vertices $s, t \in V$ and we must solve

$$\min \{ |\delta(U)| : U \subset V \text{ s.t. } s \in U, t \notin U \}.$$

This problem can be solved by network flow techniques, since the Max-Flow Min-Cut theorem tells us that the solution equals the maximum amount of flow that can be sent from s to t .

In fact, this gives us a solution to the Min Cut problem as well, because there is a reduction from the Min Cut problem to the Min s - t Cut problem. It is easy to see that the solution to the Min Cut problem equals the minimum over all pairs $s, t \in V$ of the solution to the Min s - t Cut problem. (There are more efficient reductions.)

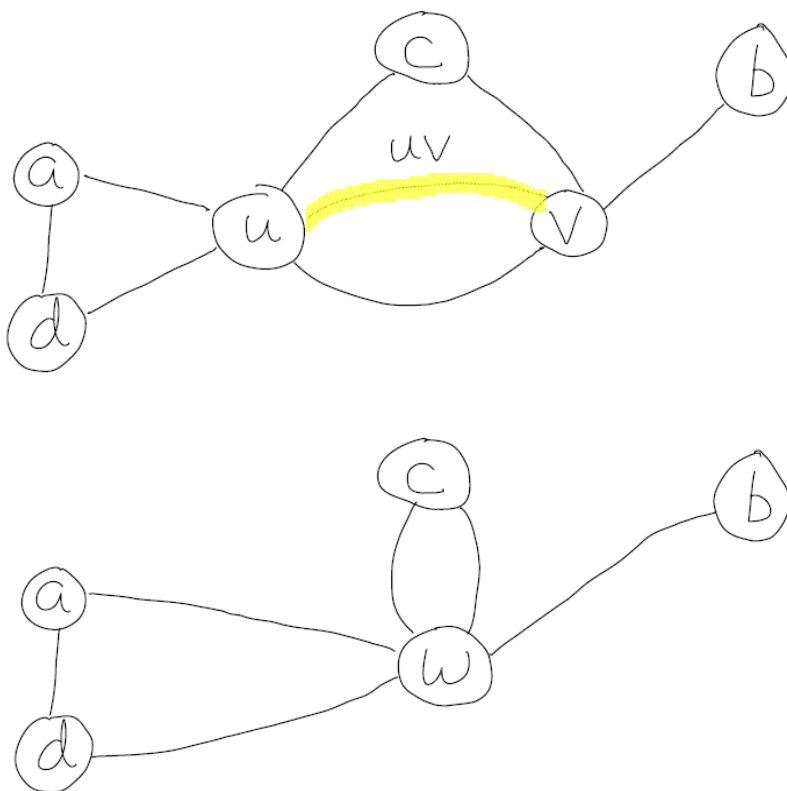
We will present a remarkable randomized algorithm for solving the Min Cut problem which does *not* use a reduction to the Min s - t Cut problem. Instead it randomly contracts edges in the graph.

1.1 Edge Contractions

Let $G = (V, E)$ be a multigraph, meaning that we allow E to contain multiple “parallel” edges with the same endpoints. Suppose that $uv \in E$ is an edge. Let us now define what it means to **contract** an edge.

Algorithm 1: Contracting an edge uv means to apply the following operations.

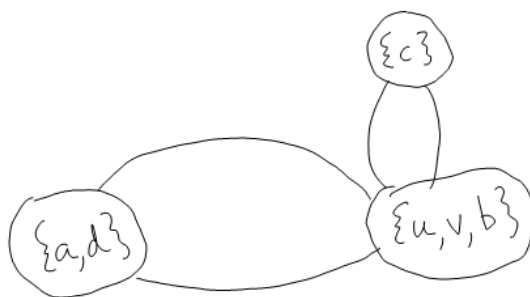
- 1 Add a new vertex w .
 - 2 For every edge xu or xv , add a new edge xw .
 - 3 This can create new parallel edges, because it might be the case that xu and xv both existed, in which case we will create *two* new edges xw .
 - 4 Every edge with an endpoint at either u or v is deleted.
 - 5 The vertices u and v are deleted.
-



Note that the contraction operation can create parallel edges, but it will not create any self-loops.

The graph that results from contracting the edge uv is written G/uv . This process essentially “merges” the two vertices u and v into a “supervertex” w which corresponds to the pair of vertices $\{u, v\}$. After performing several contraction operations, a vertex w in the contracted graph is actually a supervertex corresponding to the set of nodes that were contracted together to form w . More formally, letting $G = (V, E)$ be the original graph, each supervertex w in the contracted graph corresponds to a set of vertices $S_w \subseteq V$. These sets $\{S_w : \text{supervertex } w\}$ form a *partition* of V , meaning that they are pairwise disjoint and their union is V .

The following figure shows the result of contracting the edges uv , vb and ad . In each supervertex we show the set of vertices from the original graph that were contracted together to form the supervertex.



Claim 1 Let w be a supervertex and suppose u and v are two vertices that were contracted into w (i.e., $u, v \in S_w$). Then there is a path P between u and v in the original graph G such that every edge in P was contracted.

PROOF: This follows by induction on the number of contraction operations. \square

In the example above, u and b are in the same supervertex, and the path $u-v-b$ in the original graph had all of its edges contracted.

Claim 2 If we contract some edge uv in a graph G , then the size of a minimum cut in the contracted graph G/uv is at least the size of a minimum cut in G .

PROOF: Consider any set W of supervertices in the contracted graph. Let $U = \bigcup_{w \in W} S_w$ be the corresponding set of vertices in G . The edges between W and \bar{W} in G/uv are in bijective correspondence with the edges between U and \bar{U} in G . Therefore the size of the cut $\delta(W)$ in G/uv equals the size of $\delta(U)$ in G . So every cut in G/uv has at least as many edges as the minimum cut in G . \square

In the original example above, the minimum cut is 1 due to the cut $\delta(\{b\})$, but in the contracted example the minimum cut is 2. Consider the set $W = \{\{a, d\}\}$ containing a single supervertex. There are two edges between W and \bar{W} , and those correspond to the edges au and du in the original graph.

1.2 Computing Minimum Cuts by the Contraction Algorithm

The following randomized algorithm outputs a cut (possibly a minimum cut).

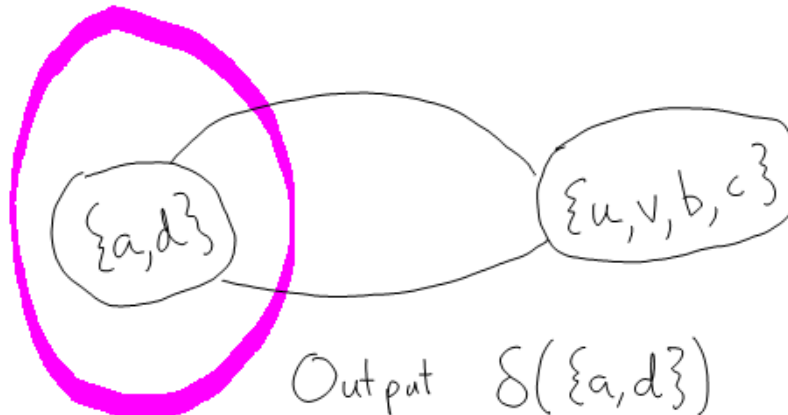
Algorithm 2: The Contraction Algorithm.

- 1 *Input:* A graph $G = (V, E)$.
 - 2 *Output:* A cut $\delta(U)$ for some non-empty set $U \subset V$.
 - 3 **while** the graph has more than two (super) vertices remaining **do**
 - 4 Pick an edge e uniformly at random
 - 5 Contract e
 - 6 Let w be one of the two remaining supervertices.
 - 7 Output the cut $\delta(S_w)$, where S_w is the set of vertices in the original graph that were contracted together to form the supervertex w .
-

To analyze this algorithm we must show that it has decent probability of outputting a minimum cut. Our main result is:

Theorem 3 Fix any minimum cut C . The contraction algorithm outputs C with probability at least $\frac{2}{n(n-1)}$.

Continuing our example above, the algorithm might decide to contract one of the edges between $\{c\}$ and $\{u, v, b\}$ (say, the edge cv in the original graph). The resulting graph is shown below. Then the algorithm outputs the cut $\delta(\{a, d\})$, which is the same as the cut $\delta(\{u, v, b, c\})$, and which contains two edges. However, this is not a minimum cut of G as the cut $\delta(\{b\})$ contains just one edge.



Before proving the theorem we need two more preliminary claims.

Claim 4 Let G be a graph with n vertices in which the minimum size of a cut is c . Then G must have at least $nc/2$ edges.

PROOF: Every vertex must have degree at least c , otherwise the edges incident on that vertex would constitute a cut of size less than c . Any graph where the minimum degree is at least c must have at least $nc/2$ edges, since the sum of the vertex degrees is exactly twice the number of edges (by the [handshaking lemma](#).) \square

Claim 5 The cut $\delta(U)$ is output by the algorithm \iff no edge in $\delta(U)$ is contracted by the algorithm.

PROOF: \Rightarrow direction: If an edge uv is contracted then the vertices u and v will belong to the same supernode from that point onwards. Therefore the set U either contains both u and v , or neither of them. In either case $uv \notin \delta(U)$.

\Leftarrow direction: Suppose no edge in $\delta(U)$ is contracted by the algorithm. Consider any pair of nodes $u \in U$ and $v \notin U$ in the original graph. If the algorithm contracts u and v into the same supernode, then Claim 1 tells us that there is a u - v path P that consists entirely of contracted edges. But this path must intersect the cut $\delta(U)$, which contradicts our hypothesis that no edge in $\delta(U)$ was contracted.

So, for every pair of nodes $u \in U$ and $v \notin U$, these two nodes belong to different supernodes in the contracted graph. At the end of the algorithm there are only two supernodes, so one of them must correspond to U and the other to its complement \bar{U} . \square

Proof (of Theorem 3). Recall that we fix an arbitrary minimum cut $C = \delta(U)$, and we must show that the algorithm has reasonable probability of outputting that particular minimum cut C . By Claim 5, this happens if and only if no edge in C is contracted. Since the contracted edges are randomly chosen, we can analyze the probability that any of those contracted edges lie in C .

Each contraction operation decreases¹ the number of vertices by one. So in the i^{th} iteration there are exactly $n - i + 1$ vertices. What can we say about the probability of contracting an edge in C during the i^{th} iteration?

Let c denote the minimum size of any cut in the original graph, so $c = |C|$. The graph in the i^{th} iteration has minimum cut size at least c , by Claim 2, and so it has at least $(n - i + 1)c/2$ edges, by Claim 4. So the probability that the randomly chosen edge in the i^{th} iteration lies in C is at most

$$\frac{|C|}{(n - i + 1)c/2} = \frac{2}{n - i + 1}.$$

Formally, let \mathcal{E}_i be the event that in the i^{th} iteration, the randomly chosen edge lies in C . Let $\overline{\mathcal{E}_i}$ be the complementary event. Unfortunately the \mathcal{E}_i 's are not independent! But this actually doesn't cause any problems. Regardless of whether the events $\mathcal{E}_1, \dots, \mathcal{E}_{i-1}$ occur or not, there are no more than c remaining edges in C , so the probability that the i^{th} random edge lies in C is at most $2/(n - i + 1)$. In particular,

$$\Pr[\overline{\mathcal{E}_i} \mid \overline{\mathcal{E}_1} \wedge \dots \wedge \overline{\mathcal{E}_{i-1}}] \geq 1 - \frac{2}{n - i + 1} = \frac{n - i - 1}{n - i + 1}.$$

So, the probability that the algorithm *never* contracts an edge in C is

$$\begin{aligned} \Pr[\overline{\mathcal{E}_1} \wedge \dots \wedge \overline{\mathcal{E}_{n-2}}] &= \prod_{i=1}^{n-2} \Pr[\overline{\mathcal{E}_i} \mid \overline{\mathcal{E}_1} \wedge \dots \wedge \overline{\mathcal{E}_{i-1}}] \\ &\geq \prod_{i=1}^{n-2} \frac{n - i - 1}{n - i + 1} \\ &\geq \frac{n - 2}{n} \cdot \frac{n - 3}{n - 1} \cdot \frac{n - 4}{n - 2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} \\ &= \frac{2}{n(n - 1)}. \end{aligned}$$

So, by Claim 5, the probability that the algorithm outputs the cut C is at least $2/n(n - 1)$. ■

Theorem 3 only proves that the algorithm has a very small probability of outputting the minimum cut C . Using the ‘‘Amplification by Independent Trials’’ technique from Lecture 2 we can boost the probability of success by performing independent trials.

Corollary 6 Fix any $\delta \in [0, 1]$. Running the contraction algorithm $n^2 \ln(1/\delta)$ times will find a minimum cut with probability at least $1 - \delta$.

After boosting the probability of success in this way, we obtain a polynomial algorithm to compute a minimum cut with high probability.

¹ This is why we defined the contraction operation so that it does not create self-loops.

1.3 Extensions

The contraction algorithm is interesting not only because it gives a simple method to compute minimum cuts, but also because there are several interesting corollaries and extensions. We now discuss a few of those.

Corollary 7 *In any undirected graph the number of minimum cuts is at most $n(n-1)/2 = \binom{n}{2}$.*

PROOF: Let C_1, \dots, C_k be the minimum cuts of the graph. Let \mathcal{E}_i be the event that C_i is output by the algorithm. Since these are disjoint events, $\sum_{i=1}^k \Pr[\mathcal{E}_i] \leq 1$. We showed above that $\Pr[\mathcal{E}_i] \geq 2/n(n-1)$ for every i , which implies that $k \leq n(n-1)/2$. \square

This bound is tight as the n -cycle has exactly $\binom{n}{2}$ minimum cuts. The next corollary proves a similar result for *approximate* minimum cuts. For any $\alpha \geq 1$, a cut is called an α -**minimum cut** if its number of edges is at most α times larger than a minimum cut.

Corollary 8 *In any undirected graph, and for any real number $\alpha \geq 1$, the number of α -minimum cuts is less than $n^{2\alpha}$.*

Finally, we state a generalization of the previous corollary which we will use next time. For every edge e let k_e denote the **edge-connectivity** of e , which is the minimum size of a cut containing e . Formally,

$$k_e := \min \{ |\delta(U)| : U \subset V \text{ and } e \in \delta(U) \}.$$

Theorem 9 *Let $G = (V, E)$ be a graph. Let $B \subseteq E$ be arbitrary and let $K = \min\{k_e : e \in B\}$. Then, for every real $\alpha \geq 1$,*

$$|\{ \delta(U) \cap B : U \subseteq V \wedge |\delta(U)| \leq \alpha K \}| < n^{2\alpha}.$$

The proof of this theorem is very similar to the proof of Corollary 8, except it needs one additional tool from graph theory called **splitting off** that we unfortunately don't have time to explain.