A Randomized Algorithms for Minimum Cuts

Andreas Klappenecker
A cut in a graph $G=(V,E)$ is a partition of the set $V$ of vertices into two disjoint sets $V_1$ and $V_2$. Edges with one end in $V_1$ and the other end in $V_2$ are said to cross the cut. A cut with a minimum number of edges crossing the cut is called a minimum cut.
Goal

Find a randomized algorithm to determine a minimum cut with high probability.
A multigraph $G = (V, E)$ is like a graph, but may contain multiple edges between vertices. Thus, $E$ is a multiset of edges rather than a set of edges.
Given a multigraph $G=(V,E)$ and an edge $e=\{C,D\}$ in $E$, the multigraph $G/e$ is obtained from $G$ by contracting the edge $e$, that is, by identifying the vertices $C$ and $D$ and removing all self-loops.

The size of the cut is given by the number of edges crossing the cut. Our goal is to determine the minimum size of a cut in a given multigraph $G$.

We describe a very simple randomized algorithm for this purpose. If $e$ is an edge of a loopfree multigraph $G$, then the multigraph $G/e$ is obtained from $G$ by contracting the edge $e=\{x,y\}$, that is, by identifying the vertices $x$ and $y$ and removing all resulting loops.

The above figure shows a multigraph $G$ and the multigraph $G/\{C, D\}$ resulting from contracting an edge between $C$ and $D$. We keep the label of each vertex to avoid cluttered notations, but keep in mind that a node $D$ in the graph $G/\{C, D\}$ really represents the set of all nodes that are identified with $D$.

Note that any cut of $G/e$ induces a cut of $G$. For instance, in the above example the cut $\{A, B\} \cup \{D,E,F\}$ in $G/\{C, D\}$ induces the cut $\{A, B\} \cup \{C, D, E, F\}$ in $G$.

In general, the vertices that have been identified in $G/e$ are in the same partition of $G$. The size of the minimum cut of $G/e$ is at least the size of the minimum cut of $G$, because all edges are kept. Thus we can use successive contractions to estimate the size of the minimum cut of $G$. This is the basic idea of the following randomized algorithm.

The algorithm Contract selects uniformly at random one of the remaining edges and contracts this edge until two vertices remain. The cut determined by this algorithm contains precisely the edges that have not been contracted. Counting the edges between the remaining two vertices yields a sample estimate of the size of the minimum cut of $G$.

The algorithm is best understood by an example. Figure 0.1 shows two different runs of the algorithm Contract. Let us have a closer look at the run...
Edge Contraction

An edge in $G$ remains in $G/e$ with the exception of the edges $e$.

If $e=\{C,D\}$, then any edge incident with $C$ or $D$ in $G$ is incident in $G/e$ with the merged node $\{C,D\}$.
A cut in $G/\{C,D\}$ leads to a cut in $G$ such that $C$ and $D$ are in the same block of the cut.

The size of the minimum cut of $G/\{C,D\}$ is at least the size of the minimum cut of $G$.

If $e=\{C,D\}$ did not cross a minimum cut, then $G/e$ has the same size minimum cut than $G$.

If $e=\{C,D\}$ crosses the minimum cut, then the size of the minimum cut of $G/e$ might be larger than the size of the minimum cut of $G$. 

Main Idea
The Randomized Algorithm

Contract\((G=(V,E))\) // G is a connected loopfree multigraph with \(|V|\geq 2\).

while \(|V|>2\) {
    Select \(e \in E\) uniformly at random;
    \(G := G/e\);
}
return \(|E|\); // \(|E|\) is an upper bound on the minimum cut of \(G\).
Example

Two different runs of the Contract algorithm. The algorithm does not always produce the correct result. The run shown in the left column correctly determines the minimum cut size to be 3. The run shown in the right column fails to produce the correct result; here the algorithm will claim that the size of the minimum cut is 6.
Example

Fig. 0.1 Two different runs of the Contract algorithm. The algorithm does not always produce the correct result. The run shown in the left column correctly determines the minimum cut size to be 3. The run shown in the right column fails to produce the correct result; here the algorithm will claim that the size of the minimum cut is 6.

/{D,F}/
Example

Two different runs of the Contract algorithm. The algorithm does not always produce the correct result. The run shown in the left column correctly determines the minimum cut size to be 3. The run shown in the right column fails to produce the correct result; here the algorithm will claim that the size of the minimum cut is 6.
Example

Fig. 0.1 Two different runs of the Contract algorithm. The algorithm does not always produce the correct result. The run shown in the left column correctly determines the minimum cut size to be 3. The run shown in the right column fails to produce the correct result; here the algorithm will claim that the size of the minimum cut is 6.
Example

Fig. 0.1 Two different runs of the Contract algorithm. The algorithm does not always produce the correct result. The run shown in the left column correctly determines the minimum cut size to be 3. The run shown in the right column fails to produce the correct result; here the algorithm will claim that the size of the minimum cut is 6.

Contractions: \{E,F\}, \{D,F\}, \{C,D\}, \{B,D\}. Cut: \{A\}, \{B,C,D,E,F\}
Intuition

Why does it work?

If a cut is of large size, then it is likely that one of its crossing edges is selected for contraction.

If a cut is of small size, then it is less likely that one of its crossing edges is selected for contraction.

⇒ Algorithm has a natural bias towards minimum cuts!
Let $C$ be one fixed minimum cut of a multigraph $G$ with $n$ vertices.

Let $E_k$ denote the event that no edge of $C$ is picked for contraction during the $k^{th}$ iteration of the algorithm.

Goal: Estimate $\Pr[E_1 \cap E_2 \cap \ldots \cap E_{n-2}] = \Pr[\text{find minimum cut } C]$
Analysis

We have \( \Pr[E \cap F] = \Pr[E|F] \Pr[F] \).

Thus, it follows that

\[
\Pr[E_{n-2} \cap E_{n-3} \cap \ldots \cap E_1] = \Pr[E_{n-2} | E_{n-3} \cap \ldots \cap E_1] \Pr[E_{n-3} \cap \ldots \cap E_1]
\]

\[
= \Pr[E_{n-2} | E_{n-3} \cap \ldots \cap E_1] \Pr[E_{n-3} | E_{n-4} \cap \ldots \cap E_1] \Pr[E_{n-4} \cap \ldots \cap E_1]
\]

\[
= \Pr[E_{n-2} | E_{n-3} \cap \ldots \cap E_1] \Pr[E_{n-3} | E_{n-4} \cap \ldots \cap E_1] \ldots \Pr[E_2 | E_1] \Pr[E_1]
\]

The conditional probabilities are not difficult to calculate!
Analysis

Suppose that the size of the minimum cut is $k$.

This means that the degree of each vertex is at least $k$, hence there exist at least $kn/2$ edges.

The probability to select an edge crossing the cut $C$ in the first step is at most $k/(kn/2) = 2/n$. Consequently, $\Pr[E_1] \geq 1 - 2/n = (n - 2)/n$. 

At the beginning of the \(m^{th}\) step, with \(m \geq 2\), there are \(n-m+1\) remaining vertices. Assuming that none of the edges crossing \(C\) were selected in previous steps, the minimum cut is still at least \(k\), hence the multigraph has at this stage at least \(k(n-m+1)/2\) edges. The probability to select an edge crossing the cut \(C\) is \(2/(n-m+1)\). It follows that

\[
\Pr[E_m|E_{m-1}\cap \ldots \cap E_1] \geq 1 - \frac{2}{(n-m+1)} = \frac{(n-m-1)}{(n-m+1)}.
\]
Conclusion

\[
\Pr \left[ \bigcap_{j=1}^{n-2} E_j \right] \geq \prod_{m=1}^{n-2} \left( \frac{n-m-1}{n-m+1} \right) = \frac{2}{n(n-1)}.
\]
Run the algorithm $a(n-1)n/2=a\left(\frac{n}{2}\right)$ times. Since $1-x \leq e^{-x}$ holds for all $x$, the probability that one of the $a$ runs finds the minimum cut is at least

$$1 - \left(1 - \frac{1}{\binom{n}{2}}\right)^a \geq 1 - e^{-a}$$

Choosing $a=c \ln n$, so a total of $c \ln(n) \binom{n}{2}$ repetitions yields

$$\Pr[\text{find minimum cut}] \geq 1 - \exp(-c \ln n) = 1 - 1/n^c.$$