Lecture Notes on *Karger's Min-Cut Algorithm.*

Eric Vigoda
Georgia Institute of Technology
Last updated for 4540 - Advanced Algorithms, Fall 2013.

Today's topic: Karger's min-cut algorithm [3].

Problem Definition

Let $G = (V, E)$ be an undirected graph. Let $n = |V|$ denote the number of vertices and $m = |E|$ denote the number of edges.

For $S \subset V$, let $\delta_G(S) = \{(u, v) \in E : u \in S, v \in \overline{S}\}$ denote the set of edges crossing between S and $\overline{S} = V \setminus S$. This set $\delta_G(S)$ is called a *cut* since their removal from G disconnects G into more than one component. We often refer to $\delta_G(S)$ as $\delta(S)$ when it is clear which graph G we are referring to.

Goal: Find the cut of minimum size. In other words, we want to find the fewest edges that need to be removed so that the graph is no longer connected (we might as well assume G is initially connected, otherwise there's nothing to do).

Previous Work

Closely related is the minimum st-cut problem. In this problem, we also take two vertices s and t as input, and we only consider cuts that separate s and t into different components, thus, we restrict attention to cuts $\delta(S)$ where $s \in S$ and $t \notin S$.

Traditionally, the min-cut problem was solved by solving $n - 1$ minimum st-cut problems. The size of the minimum st-cut is equal to the value of the maximum st-flow, and the fastest algorithm for solving maximum st-flow runs in $O(nm \log(n^2/m))$ time [1]. In fact, all $n - 1$ maximum st-flow computations can be done simultaneously with the same time bounds [2].

Karger [3] devised a simple, clever algorithm to solve the min-cut problem (without the st-condition) without using any max-flow computations. A refinement of Karger's algorithm, due to Karger and Stein [4] is also faster than the above approach using max-st-flow computations. We will present Karger's algorithm, followed by the refinement.

Algorithm Definition

The basic operation in the algorithm is to contract edges, which is defined formally below, but a rough description is for an edge $e = (v, w)$, to contract e we merge v and w into one vertex. The main idea is that if the min cut is $\delta(S)$, if both v and w are in S or both are in $\overline{S}$ (and thus e is not in the cut $\delta(S)$), then by contracting the edge e we have not changed $\delta(S)$ (it looks the same in the new graph).

The algorithm will start initially with a simple graph as input, and then it will need to consider multigraphs. In a multigraph there are possibly multiple edges between a pair of vertices. We do not have edges of the form (v, v) in our multigraphs; we refer to these types of edges as "self-loops".

Definition 1. *Let $G = (V, E)$ be a multigraph without self-loops. For $e = \{v, w\} \in E$, the* contraction *with respect to e, denoted G/e, is formed by the following steps:*

1. *Replace vertices v and w with a new vertex, z.*
2. *Replace every edge (v, x) or (w, x) with an edge (z, x).*
3. *Remove self-loops to z.*
4. *The resulting multigraph is G/e.*

As pointed out above, the key observation is that if we contract an edge (v, w) then we preserve those cuts where v and w are both in S or both in $\overline{S}$. This is formalized in the following statement.

Observation 2. *Let $e = (v, w) \in E$. There is a one-to-one correspondence between cuts in G which do not contain any edge (v, w), and cuts in G/e. In fact, for $S \subset V$ such that $v, w \in S$, $\delta_G(S) = \delta_{G/e}(S)$ (with z substituted for v and w).*

Thus, starting from G, if we contract some edges to get a multigraph G', then all of the cuts in G' correspond to cuts in G, but some cuts in G no longer are represented in G'. Let k denote the size of the minimum cut in G. A key point that we'll use later is that the size of the minimum cut in G' is $\geq k$. Why? If it's smaller, then this small cut in G' corresponds to a cut in G which is also smaller than k, contradicting our assumption that the minimum cut in G is of size k.

The idea of the algorithm is to contract $n - 2$ edges and then two vertices remain. These two remaining vertices correspond to sets of vertices in the original graph. Hence, these two vertices correspond to a partition $(S, \overline{S})$ of the original graph, and the edges remaining in this two vertex graph correspond to $\delta(S)$ in the original input graph. So we will output this cut $\delta(S)$ as our guess for the minimum cut of the original input graph. We will see momentarily why this is a good guess. But first we need to specify the algorithm more precisely, namely we need to decide:

What edges do we choose to contract in each of the $n - 2$ steps?

Consider a minimum cut $\delta(S^*)$, if there are multiple min-cut's just choose one arbitrarily. If we never contract edges from this cut $\delta(S^*)$, then, by Observation 2, this is the cut the algorithm will end up with. Since $\delta(S^*)$ is of minimum size, it has relatively few edges, so if we contract a random edge intuitively we are unlikely to choose an edge of this min-cut. It turns out that by contracting random edges we will have a reasonable probability of preserving this min-cut $\delta(S^*)$. Here is the formal algorithm.

Karger's min-cut algorithm:
Starting from the input graph $G = (V, E)$, repeat the following process until only two vertices remain:

1. Choose an edge $e = (v, w)$ uniformly at random from E.
2. Set $G = G/e$.

These final two vertices correspond to sets S, $\overline{S}$ of vertices in the original graph, and the edges remaining in the final graph correspond to the edges in $\delta(S)$, a cut of the original graph.

Note, in the algorithm we are considering a multigraph. Suppose we have a multigraph G' with m' edges, and a pair of vertices v and w have 5 edges between them in G'. Then these 5 edges are distinct, and the chance of choosing to contract (v, w) is $5/m'$.

Analysis of Algorithm's Error Probability

We need to show that Karger's min-cut algorithm has a "reasonable" chance of ending at a minimum cut of the original graph:

Lemma 3. *Let $\delta(S^*)$ be a cut of minimum size of the graph $G = (V, E)$.*

$$\Pr(\text{ Karger's algorithm ends with the cut } \delta(S^*)\ ) \geq \frac{1}{\binom{n}{2}}.$$

Proof. Fix some minimum cut $\delta(S^*)$ in the original graph. Let $|\delta(S)| = k$ denote the size of this cut.

Denote the edges we contract in the algorithm as $\{e_0, e_1, \ldots, e_{n-3}\}$. Let G_j denote the multigraph we have after j contractions. Hence, G_0 is the original input graph G, and G_{n-2} is the final graph. For $j > 0$ we have that:

$$G_j = G_{j-1}/e_j.$$

The algorithms succeeds if none of the contracted edges are in $\delta(S^*)$, in other words,

$$\text{Karger's algorithm outputs } \delta(S^*) \iff e_0, e_1, \ldots, e_{n-3} \notin \delta(S*).$$

Let us first consider the probability that the algorithm succeeds in the first contraction, i.e., that $e_0 \notin \delta(S^*)$. Note,

$$\Pr(\ e_0 \notin \delta(S^*)\ ) = 1 - \frac{|\delta(S^*)|}{|E|} = 1 - \frac{k}{m}.$$

We need to lower bound this probability, and hence we need a lower bound on m. To be useful we need to lower bound m in terms of k.

Since k is the size of the minimum cut, notice that every vertex must have degree at least k. Why? If some vertex x has degree $< k$ then the set of edges incident to x is a cut smaller than k, which contradicts our initial assumption that the size of the minimum cut is k. Since G has minimum degree $\geq k$, it has at least $nk/2$ edges (since the total degrees is $\geq nk$ and this counts each edge twice). Hence,

$$\Pr(\ e_0 \notin \delta(S^*)\ ) = 1 - \frac{k}{m} \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}. \tag{1}$$

This same approach works for analyzing later stages of the algorithm. Recall, by Observation 2, every cut in an intermediate multigraph corresponds to a cut of the original graph. Hence, for all of the multigraphs considered by the algorithm the min-cut size is $\geq k$. Otherwise, we have a vertex z in this multigraph with degree $< k$, and the edges incident this vertex correspond to a cut in this multigraph and also a cut in the original graph (by Observation 2) of size $< k$. So now let's compute the probability that the first two contractions were successful, which is $\Pr(\ e_1, e_2 \notin \delta(S^*)\ )$. Recall, Bayes formula for conditional probabilities which says that:

$$\Pr(\ e_1 \notin \delta(S^*) \,|\, e_0 \notin \delta(S^*)\ ) = \frac{\Pr(\ e_1 \notin \delta(S^*) \text{ and } e_0 \notin \delta(S^*)\ )}{\Pr(\ e_0 \notin \delta(S^*)\ )}.$$

Therefore, rearranging we have that:

$$\Pr(\ e_0, e_1 \notin \delta(S^*)\ ) = \Pr(\ e_0 \notin \delta(S^*)\ ) \Pr(\ e_1 \notin \delta(S^*) \,|\, e_0 \notin \delta(S^*)\ ) \tag{2}$$

We bounded $\Pr(\ e_0 \notin \delta(S^*)\ )$ in (1). And we can now bound $\Pr(\ e_1 \notin \delta(S^*) \,|\, e_0 \notin \delta(S^*)\ )$. The graph G_1 has $n-1$ vertices, and we just argued that it has minimum degree $\geq k$. Therefore, it has $\geq (n-1)k/2$ edges, and hence,

$$\Pr(\ e_1 \notin \delta(S^*) \,|\, e_0 \notin \delta(S^*)\ ) \geq 1 - \frac{k}{(n-1)k/2} = 1 - \frac{2}{n-1}. \tag{3}$$

Now plugging (1) and (3) back into (2) we have that:

$$\Pr(\ e_0, e_1 \notin \delta(S^*)\ ) \geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) = \frac{n-2}{n} \times \frac{n-3}{n-1}.$$

Now we can do the general computation. Recall, G_j denotes the multigraph after j contractions. The multigraph G_j contains $n - j$ vertices since we lose one vertex per contraction. Since we observed that G_j has minimum degree $\geq k$, we know that G_j has at least $(n-j)k/2$ edges.

We can now compute the probability that the algorithm successfully finds our specific minimum cut $\delta(S^*)$. To do so, all of the contracted edges must not be in $\delta(S^*)$:

$$\begin{aligned}
\Pr(\text{ final graph} = \delta(S^*)\ ) &= \Pr(\ e_0, e_1, \ldots, e_{n-3} \notin \delta(S^*)\ ) \\
&= \Pr(\ e_0 \notin \delta(S^*)\ ) \Pr(\ e_1, \ldots, e_{n-3} \notin \delta(S^*) \,|\, e_0 \notin \delta(S^*)\ ) \\
&= \Pr(\ e_0 \notin \delta(S^*)\ ) \prod_{j=1}^{n-3} \Pr(\ e_j \notin \delta(S^*) \,|\, e_0, \ldots, e_{j-1} \notin \delta(S^*)\ ) \\
&\geq \prod_{j=0}^{n-3} \left(1 - \frac{k}{(n-j)k/2}\right) \\
&= \prod_{j=0}^{n-3} \left(1 - \frac{2}{n-j}\right) \\
&= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \frac{n-5}{n-3} \cdots \times \frac{3}{5} \times \frac{2}{4} \times \frac{1}{3} \\
&= \frac{(2)(1)}{(n)(n-1)} \\
&= \frac{1}{\binom{n}{2}}.
\end{aligned}$$

□

Boosting the Success Probability

We'd like to boost the probability of success, so that the algorithm succeeds with high probability, let's say with probability at least $1 - 1/n^{-10}$. In fact, we'll see that we can achieve any inverse polynomial for the error probability by simply changing the constants in the following scheme.

To boost the success probability, we simply run the above algorithm $10n^2 \ln(n)$ times, and we output the best of the cuts we see. Before analyzing this scheme, a simple fact – recall, the Taylor's expansion for $\exp(-x) = 1 - x + x^2/2 \pm \cdots$ where $x \geq 0$. Hence, for $0 \leq x \leq 1$ we have that: $\exp(-x) \geq 1 - x$. Fix a minimum cut $\delta(S^*)$. The probability that all $10n^2 \ln(n)$ runs of the algorithm fail to output $\delta(S^*)$ is at most

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{10n^2 \ln(n)} \leq \exp\left(-\frac{10n^2 \ln(n)}{\binom{n}{2}}\right) \leq \exp(-10 \ln(n)) = n^{-10}.$$

Therefore, the probability we find a minimum cut is at least $1 - n^{-10}$, as desired.

Running Time

It's easy to implement Karger's algorithm so that one run takes $O(n^2)$ time – that will be your first homework problem. Therefore, we have an $O(n^4 \log n)$ time randomized algorithm with error probability $1/\text{poly}(n)$.

A faster version of this algorithm was devised by Karger and Stein [4]. The key idea comes from looking at the telescoping product. In the initial contractions it's very unlikely we contracted an edge in the minimum cut. Towards the end of the algorithm, our probability of contracting an edge in the minimum cut grows.

From the earlier analysis we have the following. For a fixed minimum cut $\delta(S)$, the probability that this cut survives down to ℓ vertices is at least $\binom{\ell}{2}/\binom{n}{2}$. Thus, for $\ell = n/\sqrt{2}$ we have probability $\geq 1/2$ of succeeding. Hence, in expectation two trails should suffice.

Improved algorithm: From a multigraph G, if G has at least 2 vertices, repeat twice:

1. run the original algorithm down to $n/\sqrt{2}$ vertices.
2. recurse on the resulting graph.

Return the minimum of the cuts found in the two recursive calls.

We can easily compute the running time via the following recurrence (which is straightforward to solve, e.g., the standard Master theorem applies):

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

Since we succeed down to $n/\sqrt{2}$ with probability $\geq 1/2$, we have the following recurrence for the probability of success, denote by $P(n)$:

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

This solves to $P(n) = \Omega\left(\frac{1}{\log n}\right)$. By a similar boosting argument as we used for the original algorithm, with $O(\log^2 n)$ runs of the algorithm, the probability of success is $\geq 1 - 1/poly(n)$.

Therefore, in $O(n^2 \log^3 n)$ total time, we can find the minimum cut with probability $\geq 1 - 1/poly(n)$.

Number of Minimum Cuts

Before finishing, we observe an interesting corollary of Karger's original algorithm.

Corollary 4. *Any graph has at most $O(n^2)$ minimum cuts.*

Proof. Consider the collection of all cuts of minimum size, and denote this collection by $S_1, \ldots, S_j$. For all $1 \leq i \leq j$, the probability that one run of Karger's algorithm finds S_i is $\geq 1/n^2$. Since only cut is outputted by each run of Karger's algorithm we must have that $j \leq n^2$. □

Note, we can also enumerate all of these cuts by the above algorithm.

References

[1] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921-940, 1988.

[2] J. Hao and J. B. Orlin. A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 165-174, 1992.

[3] D. R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 21-30, 1993.

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