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

Eric Vigoda
Georgia Institute of Technology
Last updated for 7530 - Randomized Algorithms, Spring 2010. [1]

Today's topic: Karger's min-cut algorithm.

Let $G = (V, E)$ be an undirected graph. Let $n = |V|, m = |E|$.

For $S \subset V$, the set $\delta(S) = \{(u, v) \in E : u \in S, v \in \overline{S}\}$ is a *cut* since their removal from G disconnects G into more than one component.

Goal: Find the cut of minimum size.

Closely related is the minimum st-cut problem. In this problem, for specified vertices s and t we restrict attention to cuts $\delta(S)$ where $s \in S$, $t \notin S$.

Traditionally, the min-cut problem was solved by solving $n - 1$ min-st-cut problems. In the min-st-cut problem we are given as input two vertices s and t, our aim is to find the set S where $s \in S$ and $t \notin S$ which minimizes the size of the cut $(S, \overline{S})$, i.e., $|\delta(S)|$. The size of the min-st-cut is equal to the value of the max-st-flow (equivalent by linear programming duality). The fastest algorithm for solving max-st-flow runs in $O(nm \log(n^2/m))$ time [1]. In fact, all $n - 1$ max-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. An 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.

The basic operation in the algorithm is to contract edges, which is defined below. 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 $e = \{u, v\} \in E$, the* contraction *with respect to e, denoted G/e, is formed by:*

1. *Replace vertices u and v with a new vertex, w.*
2. *Replace all edges (u, x) or (v, x) with an edge (w, x).*
3. *Remove self-loops to w. G/e is a multigraph.*

The key observation is that if we contract an edge (u, v) then we preserve those cuts where u and v are both in S or both in $\overline{S}$.

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

The idea of the algorithm is to contract $n - 2$ edges and then two vertices remain. These two vertices correspond to a partition $(S, \overline{S})$ of the original graph, and the edges remaining in the two vertex graph

[1] Based on scribe notes first prepared by Tom Hayes at the University of Chicago in the winter quarter, 2003.

correspond to $\delta(S)$ in the original input graph. So we will output this cut $\delta(S)$ as what think is the minimum cut of the original input graph. What edges do we contract? If we never contract edges from a minimum 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 it turns out that 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 $e = (u, v)$ 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.

Claim: This process has a reasonable chance of ending at a minimum cut of the original graph.

Fix some minimum cut $\delta(S)$ in the original graph. Let $|\delta(S)| = k$.

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

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

Proof: Denote the edges we contract in the algorithm as $\{e_1, e_2, \ldots, e_{n-2}\}$. The algorithms succeeds if none of the contracted edges are in $\delta(S)$. To upper bound the probability that the algorithm succeeds in the first contraction, i.e., that $e_1 \notin \delta(S)$ we need to lower bound the number of edges in the input graph G in terms of k. Note, the minimum degree is at least k in G, otherwise we have a cut of size smaller than k since we can disconnect the vertex from the rest of the graph by removing all edges incident to it. This implies that the original input graph G has at least $nk/2$ edges. By Observation 2, since every cut in an intermediate multigraph corresponds to a cut of the original graph, we have the following observation.

Observation 4 *The minimum degree in all of the intermediate multigraphs is at least k. Otherwise, the edges incident the (meta)vertex with degree smaller than k would correspond to a cut of size $< k$ in the original graph.*

After j contractions, the multigraph, denote as G_j, contains $n - j$ vertices since we lose one vertex per contraction. Then, Observation 4 implies that G_j has at least $(n - j)k/2$ edges.

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

$$\begin{aligned}
\Pr(\text{ final graph} = \delta(S)\ ) &= \Pr(\ e_1, e_2, \ldots, e_{n-2} \notin \delta(S)\ ) \\
&= \Pr(\ e_1 \notin \delta(S)\ ) \prod_{j=1}^{n-3} \Pr(\ e_{j+1} \notin \delta(S) | e_1, \ldots, e_j \notin \delta(S)\ ) \\
&\geq \prod_{j=0}^{n-3} \left(1 - \frac{k}{(n-j)k/2}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \cdots \times \frac{2}{4} \times \frac{1}{3} \\
&= \frac{2}{(n)(n-1)} \\
&= \frac{1}{\binom{n}{2}}.
\end{aligned}
$$

In order to boost the probability of success, we simply run the algorithm $\ell\binom{n}{2}$ times. The probability that at least one run succeeds is at least

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

Setting $\ell = c \ln n$ we have error probability $\leq 1/n^c$. ■

It's easy to implement Karger's algorithm so that one run takes $O(n^2)$ time. 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 6 vertices, repeat twice:

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

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

The choice of 6 as opposed to some other constant will only affect the running time by a constant factor.

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

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

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 $P(n)$:

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

This solves to $P(n) = \Omega\left(\frac{1}{\log n}\right)$. Hence, similar to the earlier argument 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 $\geq 1 - 1/poly(n)$.

Before finishing, we observe an interesting corollary of Karger's original algorithm which we will use in the next lecture to estimate the (un)reliability of a network.

Corollary 5 *Any graph has at most $O(n^2)$ minimum cuts.*

This follows from Lemma 3 since that holds for any specified minimum cut.

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. *J. Assoc. Comput. Mach.*, 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 Third Annual ACM-SIAM Symposium on Discrete Algorithms (Orlando, FL, 1992)*, pages 165–174, New York, 1992. ACM.

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

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