

Lecture 1: January 7, 2003

Lecturer: Eric Vigoda

Scribe: Tom Hayes

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

Let $G = (V, E)$ be an undirected graph. For $S \subset V$, the set $\delta(S) = \{(u, v) \in E : u \in S, v \in \bar{S}\}$ is a *cut*.

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. The size of the min-st-cut is equal to the value of the max-st-flow (equivalent by linear programming duality). Let $n = |V|, m = |E|$. 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].

Open problem?: Solve min-st-cut without using max st-flow.

But, if we drop the $s - t$ condition, there is an easy algorithm to do just this. The algorithm we begin with was devised by Karger [3].

Definition 1.1 Let $G = (V, E)$ be a multigraph without self loops. For $e = \{u, v\} \in E$, the contraction with respect to $e, G/e$, is:

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.

Observation 1.2 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).

Karger's min-cut algorithm: From a multigraph $G = (V, E)$, choose an edge $e = (u, v)$ uniformly at random from E . Set $G = G/e$. Repeat this process until G has only 2 vertices. These final vertices correspond to sets S, \bar{S} of vertices in the original graph, and the final edges correspond to the edges in $\delta(S)$, a cut.

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

Consider some minimum cut $\delta(S)$ in the original graph, say $|\delta(S)| = k$. Denote the edges we contract in the algorithm as $\{e_1, e_2, \dots, e_{n-2}\}$.

Observation: The minimum degree in all 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 $< k$.

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 $\delta(S)$.

$$\begin{aligned} \Pr(\text{final graph} = \delta(S)) &= \prod_i \Pr(e_i \notin \delta(S)) \\ &\geq \prod_{j=0}^{n-3} \left(1 - \frac{k}{(n-j)k/2}\right) \\ &= \frac{n-2}{n} \frac{n-3}{n-1} \cdots \frac{2}{4} \frac{1}{3} \\ &= \frac{1}{\binom{n}{2}}. \end{aligned}$$

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

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

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 trials 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.

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

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 $P(n) = \Omega\left(\frac{1}{\log n}\right)$. By $O(\log n)$ runs of the algorithm, the probability of success is $\geq 1 - 1/\text{poly}(n)$.

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

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

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

We conclude with a proposal for an alternative algorithm to find a minimum cut. The goal is not necessarily to compete with the linear time algorithm, but rather to find a somewhat simpler, more natural process which may be useful in the next lecture for network (un)reliability.

Alternative algorithm? What if we do some sort of random walk in which sometimes we contract a random edge, and sometimes we uncontract the last edge we contracted. During this walk we keep track of the minimum degree we've encountered, and output this as the minimum cut. The probabilities of contracting versus uncontracting will of course depend on the size of the current multigraph; If we have many vertices the probability of contracting should be higher than if we have few vertices remaining. How well does such a random walk perform?

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.