PRINCETON COS 521: ADVANCED ALGORITHM DESIGN

Randomized Minimum Cut

TODAY'S TOPIC is simple but gorgeous: Karger's min-cut algorithm [1] and its extension. It is a simple randomized algorithm for finding the *global minimum cut* in an undirected graph: a (non-empty) subset of vertices X in which the set of edges leaving S, denoted $E(X, \overline{X})$ has minimum size among all subsets. You may have seen an algorithm for this problem in your undergrad class that uses maximum flow. Karger's algorithm is elementary and a great introduction to randomized algorithms. For simplicity, we will consider unweighted graphs (the algorithm also works for weighted graphs).

[1] David 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*, SODA '93, page 21–30, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0898713137

Basic Operations

Karger's algorithm makes use of the following basic operations.

1. **Select a random edge:** For this lecture (and most lectures), we won't stress about how the algorithm accomplishes this. For instance, the algorithm might have the ability to select a random number between 1 and $m = |E|$, and can select the edge indexed by the random number. We will assume for the purpose of this class that the algorithm can select a random edge in time $O(1)$.

2. **Contract an edge:** This operation takes two existing nodes in the graph with an edge between them and "merges" them into a super-node. That is, the operation Contract($G = (V, E), e = \{u, v\}$) takes as input a graph G with vertex set V and edges E, and outputs a new graph G' with vertex set $V \setminus \{u, v\} \cup S_{u,v}$. Thus, it replaces the two nodes u and v with a supernode $S_{u,v}$. The new edge set contains all edges between two nodes in $V \setminus \{u, v\}$. For all edges between u and $x \notin \{u, v\}$, add an edge between $S_{u,v}$ and x. Ditto for all edges between v and $x \notin \{u, v\}$. **Note that this may create multiple edges between two nodes, which is intended.** But do not include edges from $S_{u,v}$ to itself (so no self-loops). In this lecture, we will also not stress about how exactly to implement

this operation, but note that it can be done in time $O(n)$ (say, if the graph is described to you in the adjacency list format).

The (Core) Algorithm

The algorithm is straightforward: Pick a random edge and contract it. Repeat until the graph has only two supernodes, which is output as our guess for min-cut. That is if, upon termination, the two remaining nodes are S_X and $S_{\bar{X}}$, output $(X, \bar{X})$ as the guess for the minimum cut.

To guarantee a high success probability, re-run the algorithm from scratch k times independently, and output whichever guess $(X_1, \bar{X}_1), \ldots, (X_k, \bar{X}_k)$ is the most minor cut. k will be chosen shortly.

Intuition

Before we get into a formal proof (itself quite simple), here is some brief intuition. We say that a given cut $(X, \bar{X})$ *survives* contraction $e = \{u, v\}$ if $|X \cap \{u, v\}| \neq 1$. That is, a cut survives the contraction of edge e as long as edge e is between two nodes on the same side of the cut. The idea is that once an edge that crosses the cut $(X, \bar{X})$ is contracted, we have guaranteed that we cannot possibly output cut $(X, \bar{X})$ at the end. On the other hand, if we never contract an edge that crosses cut $(X, \bar{X})$, then $(X, \bar{X})$ will be exactly the cut we output.

So, the idea is that we output a cut if and only if it survives all $n - 2$ contractions. At every step, the cut that is most likely to survive is the global min-cut, precisely because it has fewer edges that "kill" it than all other cuts.

This algorithm also looks like a great heuristic to try on all kinds of real-life graphs where one wants to *cluster* the nodes into “tightly-knit”portions. For example, social networks may cluster into communities, graphs capturing the similarity of pixels may cluster to give different portions of the image (sky, grass, road, etc.). Thus, instead of continuing Karger’s algorithm until you have two supernodes left, you could stop it when there are k supernodes and try to understand whether these correspond to a reasonable clustering.

Analysis

We begin with the following observation, which is more of a definition than observation:

Observation 1. *Let G' be obtained by a sequence of edge contractions of G. Then there is a one-to-one correspondence between cuts $(Y, \bar{Y})$ of G' and cuts $(X, \bar{X})$ of G that survived all contractions (note that this is an injection*

from cuts in G' to cuts in G but not a bijection). Namely, the cut $(Y, \bar{Y})$ in G' corresponds to the cut $X = \cup_{y \in Y} S(y)$, where $S(y)$ denotes the original vertices of G that were contracted to form the super-node y in G'.

The key corollary in the analysis of Karger's algorithm follows the following simple lemma:

Lemma 1. *Let G be an undirected graph, potentially with multi-edges but not self-loops, and let c be the value of the min-cut of G. Then $|E(G)| \geq nc/2$.*

Proof. The cut $(\{v\}, E \setminus \{v\})$ is a potential min-cut, and has value exactly $d(v)$ (the degree of v). Therefore, $d(v) \geq c$ for all v. We can write $|E| = \sum_v d(v)/2 \geq nc/2$. □

Corollary 1. *Let G be an undirected graph, potentially with multi-edges but not self-loops. Let $(X, \bar{X})$ be any minimum cut of G. Then, the probability that $(X, \bar{X})$ survives the contraction of a random edge is at least $(1 - 2/n)$.*

Proof. By Lemma 1, there are at least $cn/2$ edges that might be selected. Exactly c of them would kill $(X, \bar{X})$. So the probability that $(X, \bar{X})$ survives is at least $1 - \frac{c}{cn/2} = 1 - 2/n$. □

Now we can conclude with the main theorem:

Theorem 1. *[Karger 1993] For any graph G, and any min-cut $(X, \bar{X})$ of G, Karger's algorithm outputs $(X, \bar{X})$ with probability at least $\frac{2}{n(n-1)}$.*

Proof. We know that Karger's algorithm outputs $(X, \bar{X})$ if and only if X survives every contraction. The probability that X survives the first contraction is $1 - 2/n$ by Corollary 1, and the probability that it survives the i^{th} contraction, conditioned on surviving the first $i - 1$ is $1 - 2/(n - i - 1)$ (also by Corollary 1). So the probability that it survives every contraction is at least:

$$\prod_{i=1}^{n-2}(1 - 2/(n - (i-1))) = \prod_{i=1}^{n-2}(n - i - 3)/(n - i - 1) = \frac{2}{n(n-1)}.$$

The last equality is due to a telescoping product. Every numerator "cancels" with a denominator two steps down the way, so the last two numerators and the first two denominators remain. □

Each iteration succeeds in finding a minimum cut only with probability $\sim 2/n^2$ which seems honestly terrible. But there's a simple fix: repeat the algorithm with independent randomness k times and take the minimum cut of all the cuts output in the k iterations.

Lemma 2. *The chance that the algorithm fails to output a particular min cut $cut(X)$ in all of the k iterations is at most $(1-2/n^2)^k$. Thus, if $k \geq O(n^2 \ln 1/\epsilon)$, then the algorithm succeeds in outputting a fixed min cut with probability at least $1-\epsilon$.*

The proof of this lemma is immediate. The last line is based on a useful inequality: $1+x \leq e^x$ for every x. Thus, $(1-2/n^2)^k \leq e^{-2k/n^2}$.

Thus, to guarantee that the min-cut survived at least one iteration except with probability ϵ, we would need to repeat the procedure independently $\Theta(n^2 \ln(1/\epsilon))$ times. The runtime of each iteration is $O(n^2)$ because we do $n-2$ contractions, each of which takes time $O(n)$, so the total runtime of the algorithm is $O(n^4)$, which is good (poly time!) but sad.

There is a non-algorithmic consequence that we next point out that is a surprising outcome of the analysis above:

Lemma 3. *The number of min cuts in any graph is at most $\binom{n}{2}$.*

Proof. Consider the set of all min cuts. For each cut, say $(X, \overline{X})$ in this list, Theorem 1 says that the random contraction algorithm outputs $(X, \overline{X})$ with probability $2/n(n-1)$. For different cuts, these are mutually exclusive events. Thus, if the number of min cuts is t then $2t/n(n-1) \leq 1$ or $t \leq \binom{n}{2}$. □

At first blush, this is a somewhat surprising result. The total number of (non-empty/non-full) cuts in any graph is 2^{n-1}. But the above lemma says that at most $\sim n^2$ can be minimum cuts!

We will see a surprising application of this purely combinatorial result later in this course: *every graph G can be approximated by a weighted graph H with $O(nlogn)$ edges so that the weighted size of any cut in H is a $(1 \pm \epsilon)$-approximation to the same cut in G.* That is, for any problem that has to do with cuts, you can replace a dense graph by a super sparse graph – speeding up algorithms along with many other applications. Such results (called graph sparsification [2]) form an exciting area of research, continuing to this day, and have deep connections to problems in mathematics [3].

Improved Karger-Stein Algorithm

Karger and Stein [4] improve the algorithm to run in time $O(n^2 \log^2(n))$ (essentially replacing two factors of n with $\log(n)$ instead). The idea is roughly that *repetition ensures fault tolerance*. The real-life advice of making two backups of your hard drive is related to this: the probability that both fail is much smaller than one does. In the case of Karger's algorithm, the overall probability of success is too low.

[2] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in Õ(n2) time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 47–55, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. DOI: 10.1145/237814.237827. URL https://doi.org/10.1145/237814.237827

[3] e.g., to the construction of Ramanujan expanders and the resolution of the Kadison-Singer problem in quantum information, among others.

Adam Marcus, Daniel Spielman, and Nikhil Srivastava. Interlacing families i: Bipartite Ramanujan graphs of all degrees. *Annals of Mathematics*, pages 307–325, July 2015. DOI: 10.4007/annals.2015.182.1.7

[4] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, jul 1996. ISSN 0004-5411. DOI: 10.1145/234533.234534. URL https://doi.org/10.1145/234533.234534

The main idea is this: we're unlikely to kill the min-cut in the first several contractions, so why are we repeating these every time? Instead, we should be more clever about exactly which contractions we repeat. In other words, the chance of a min-cut being killed in a contraction is higher if the number of vertices in the graph is lower. We will now set up a recursive algorithm that ends up doing repetitions in a way that effectively amounts to doing more repetitions for graphs on a smaller number of vertices.

The formalization of this idea begins with the following observation about the analysis of the telescoping product in the proof of Theorem 1.

Observation 2. *Let G be an undirected graph with n nodes, possibly with multi-edges but not self-loops. Let $(X, \bar{X})$ be any minimum cut of G, then the probability that $(X, \bar{X})$ survives $n - n/\sqrt{2}$ random contractions is at least $1/2$.*

Proof. Precisely the same math as the proof of Theorem 1, except stop at $n - n/\sqrt{2}$ instead of $n - 2$. This telescopes to at least $1/2$. □

A recursive algorithm: Now, consider the following recursive algorithm: starting from a graph G (with n nodes, multi-edges but no self-loops), twice, and independently, randomly contract edges until only $n/\sqrt{2}$ nodes remain, and call the resulting graphs G'_1, G'_2. Then, call the algorithm independently on each of G'_1, G'_2, and output the smaller two returned cuts.

The runtime of the algorithm satisfies the recurrence:

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

$T(n) = O(n^2 \log n)$ solves the recurrence.[5] So, each independent run of Karger-Stein has a total runtime barely more than Karger's algorithm, but the redundancy should guarantee a higher success probability. Again, we need to analyze the probability that the min-cut will survive the Karger-Stein algorithm.

[5] Hush, hush, dont tell anybody, but most researchers dont use the Master theorem, even though it was stressed a lot in undergrad algorithms. Most researchers just "unfold" the recursion. More formally, you could instead write a binary tree, observing that there are $O(\log n)$ levels and that each level has total "excess" work of $O(n^2)$. On level i, there are 2^i nodes, each with "excess work" $O((n/\sqrt{2}^i)^2) = O(n^2/2^i)$.

Theorem 2 (Karger-Stein 1996)**.** *Let c be the value of the minimum cut of G. Then the probability that Karger-Stein outputs a cut of value c is $\Omega(1/\log n)$.*

Proof. This time, we can't restrict attention to a specific min-cut $(X, \bar{X})$ because even if that min-cut survives one of the two recurrences, the other recurrence might output a different minimum cut (and only one can be selected in the end).

So, define $P(n)$ as the probability that an iteration of Karger-Stein selects a minimum cut in graphs of size n. We see that Karger-Stein selects a minimum cut if and only if for one of the two recurrences:

- A min-cut survives the first $n - n/\sqrt{2}$ contractions (this occurs with probability at least $1/2$, by Observation).
- The recursive call succeeds. This happens with probability at least $P(n/\sqrt{2})$, by definition.

So the probability of success for each try is at least $P(n/\sqrt{2})/2$. The probability that both fail is then at most $(1 - P(n/\sqrt{2})/2)^2$, and we get $P(n) \geq 1 - (1 - P(n/\sqrt{2})/2)^2$. The last step is again solving the recurrence, which is $\Omega(1/\log n)$.

To see this, assume that $P(n/\sqrt{2}) \geq \Delta/\log n$ for some constant $\Delta > 0$. Then we get:

$$P(n) \geq 1 - (1 - \frac{\Delta}{2\log(n/\sqrt{2})})^2 = 1 - (1 - \frac{\Delta}{2\log(n) - 1})^2 \geq \frac{\Delta}{\log(n) - 1/2} - (\frac{\Delta}{2\log(n) - 1})^2$$

$$\geq \frac{\Delta}{\log(n)} + \frac{\Delta/2}{\log(n)(\log(n) - 1/2)} - \frac{\Delta^2}{(2\log(n) - 1)^2} \geq \frac{c}{\log n} + \frac{2\Delta\log(n) - \Delta - \Delta^2\log n}{\log(n) \cdot (2\log(n) - 1)^2} \geq \Delta/\log n.$$

The last inequality is true whenever $\Delta \leq 2$. So we get that the probability of success is $\Omega(1/\log(n))$. □

We can repeat the entire algorithm independently $O(\log(n)/\epsilon)$ times to get a total success rate of $1 - \epsilon$ as before.

Bibliography

András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in Õ(n2) time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 47–55, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. DOI: 10.1145/237814.237827. URL https://doi.org/10.1145/237814.237827.

David 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*, SODA '93, page 21–30, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0898713137.

David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, jul 1996. ISSN 0004-5411. DOI: 10.1145/234533.234534. URL https://doi.org/10.1145/234533.234534.

Adam Marcus, Daniel Spielman, and Nikhil Srivastava. Interlacing families i: Bipartite Ramanujan graphs of all degrees. *Annals of Mathematics*, pages 307–325, July 2015. DOI: 10.4007/annals.2015.182.1.7.