CSC473: Advanced Algorithm Design Winter 2020

Week 2: The Karger-Stein Min Cut Algorithm

Aleksandar Nikolov

In these notes we describe the Karger-Stein algorithm, which is a method to amplify the probability of success of Karger's Contraction algorithm (described in the first week of class) without slowing it down much.

Let us, again, fix an undirected, unweighted graph $G = (V, E)$ with n nodes and m vertices. We have shown that the Contraction algorithm outputs any fixed global min cut $E(S, \bar{S})$ with probability at $\frac{2}{n(n-1)}$. In order to get an algorithm with constant probability of success, we can then run the contraction algorithm $\Theta(n^2)$ times, and output the minimum cut found by any of these runs. We have also shown that a single execution of the Contraction algorithm can be implemented in time $O(n^2)$. So, this gives an algorithm with constant probability of success and running time $O(n^4)$.

In the these notes, we will show an improved algorithm by Karger and Stein, which achieves running time $O(n^2 \log^2 n)$. The main idea is to use the fact that the early contractions in the Contraction algorithm are much less likely to destroy the min cut.

Let us fix any global minimum cut $E(S, \bar{S})$, where S is a set which is not empty, and is not equal to V, and $E(S, \bar{S})$ is the set of edges with exactly one endpoint in S. In class we introduced the notation $\mathcal{E}_i$ for the event that no edge of $E(S, \bar{S})$ is contracted at the i-th contraction. We also showed that

$$\mathbb{P}(\mathcal{E}_i \mid \mathcal{E}_1 \text{ and } \mathcal{E}_2 \text{ and } \ldots \text{ and } \mathcal{E}_{i-1}) \geq \frac{n-i-1}{n-i+1}.$$

Exercise 1. *Show that the probability that no edges of $E(S, \bar{S})$ are contracted during the first i contractions satisfies*

$$\mathbb{P}(\mathcal{E}_1 \textit{ and } \mathcal{E}_2 \textit{ and ... and } \mathcal{E}_i) \geq \frac{(n-i)(n-i-1)}{n(n-1)}. \tag{1}$$

More generally, show that for any $1 \leq i < j \leq n-2$

$$\mathbb{P}(\mathcal{E}_i \textit{ and } \mathcal{E}_{i+1} \textit{ and ... and } \mathcal{E}_j \mid \mathcal{E}_1 \textit{ and } \ldots \textit{ and } \mathcal{E}_{i-1}) \geq \frac{(n-j)(n-j-1)}{(n-i+1)(n-i)}. \tag{2}$$

The important thing to notice about (1) is that the probability is at least a constant for a constant fraction of the values of i. For example, for $i \leq n - \frac{n}{\sqrt{2}} - 1$, (1) gives

$$\mathbb{P}(\mathcal{E}_1 \text{ and } \mathcal{E}_2 \text{ and ... and } \mathcal{E}_i) \geq \frac{n^2}{2n(n-1)} \geq \frac{1}{2}.$$

Similarly, suppose that we have made $i-1$ contractions, and currently have $\ell = n-i+1$ remaining (super-)nodes. Then, by (2), after $j = \ell - \frac{\ell}{\sqrt{2}} - 1$ additional contractions

for $j \leq n - \frac{\ell}{\sqrt{2}} - 1$, we have

$$\mathbb{P}(\mathcal{E}_i \text{ and } \mathcal{E}_{i+1} \text{ and ... and } \mathcal{E}_{i+j-1} \mid \mathcal{E}_1 \text{ and } \ldots \text{ and } \mathcal{E}_{i-1}) \geq \frac{\ell^2}{2\ell(\ell-1)} \geq \frac{1}{2}.$$

The main idea in Karger and Stein's improved algorithm is to stop the Contraction algorithm after this many contractions, while the probability that no mistake was made is still large, and repeat *not* from the start, but from the current graph. This way later contractions, which are more likely to lead to an error, are repeated more often than the early ones, which are unlikely to lead to an error.

MIN-CUT(G)

// G can be a multigraph, with supernodes corresponding to sets of vertices.
1 **if** G has two supernodes corresponding to $S, \bar{S}$
2 Output $(S, \bar{S})$
3 Run the Contraction algorithm until $\frac{n}{\sqrt{2}} + 1$ supernodes remain.
4 Let G' be the resulting contracted multigraph.
5 $(S_1, \bar{S}_1) =$ MIN-CUT(G').
6 $(S_2, \bar{S}_2) =$ MIN-CUT(G'). // The two recursive runs of MIN-CUT are independent.
7 Output the cut $(S_i, \bar{S}_i)$ ($i \in \{1, 2\}$) with smaller $|E(S_i, \bar{S}_i)|$.

Theorem 1. *The algorithm* MIN-CUT(G) *runs in time* $O(n^2 \log n)$ *and outputs a global min cut of* G *with probability at least* $\frac{1}{O(\log n)}$.

Proof. Let us first analyze the running time, which is easier. The $n - \frac{n}{\sqrt{2}} - 1$ contractions take time $O(n^2)$ using the implementation using adjacency matrices described in class. Then we make two recursive calls on a multigraph with $\frac{n}{\sqrt{2}} + 1$ nodes. So, we have the recursion

$$T(n) = 2T\left(\frac{n}{\sqrt{2}} + 1\right) + O(n^2).$$

for the worst-case running time $T(n)$.

Exercise 2. *Verify that the recursion above solves to* $T(n) = O(n^2 \log n)$.

The hard part is to analyze the probability of success. Here it helps to imagine the recursion tree. It is just a binary tree, where each edge corresponds to one recursive call to MIN-CUT(G'), and each node corresponds to one partial run of the Contraction algorithm. The height of the tree is $O(\log n)$, because after each level we recurse on a graph with approximately $\frac{n}{\sqrt{2}}$ nodes.

Let us say that a node of the tree is *successful* if there is some min cut of the original graph, so that no edge of the min cut is contracted during the partial run of the Contraction algorithm that corresponds to that node. By design, conditional on all ancestors of a node being successful, the node itself is successful with probability at least $\frac{1}{2}$. The algorithm overall succeeds in finding a min cut if there is path from the root to a leaf in the recursion tree that contains only successful nodes. We call a path containing only successful nodes a *successful path.*

So, we need to analyze the probability that the recursion tree contains a successful path from the root to a leaf. Let us pose this problem abstractly. Let T be a complete binary tree of height h. Suppose that for every node u of T, the probability that u is successful, conditional on the ancestors of u being successful, is exactly $\frac{1}{2}$. Moreover, assume that for every node u, the subtrees below the left and the right child of u are independent. Let $P(d)$ be the probability that there is a successful

path from the root to some node at distance d from the root of T. $P(h)$ is a lower bound on the probability that Min-Cut(G) outputs a min cut of G.

To finish the proof, we analyze $P(d)$. By definition, $P(0) = \frac{1}{2}$. Also, $P(d)$ equals

$$P(d) = \mathbb{P}(\text{the root is successful}) \times \\ \mathbb{P}(\text{there is a successful path from a node of depth 1 to a node at depth } d \mid \text{the root is successful}),$$

where we can multiply the two probabilities because the two events are independent. The first probability is $\frac{1}{2}$. Note that, conditional on the root being successful, the probability that there is a successful path from the left child of the root to a node at depth d is $P(d-1)$, and similarly for the right child of the root: the subtrees under the two children are identically distributed to the first $h-1$ levels of T. Also, the event that there is such a path from the left child and the event that there is such a path from the right child are independent. Then the probability that neither of these events holds is $(1 - P(d-1))^2$, and we have

$$P(d) \geq \frac{1}{2}(1 - (1 - P(d-1))^2) = P(d-1) - \frac{1}{2}P(d-1)^2.$$

Exercise 3. *Verify that the function $f(p) = p - \frac{1}{2}p^2$ is a non-decreasing function of p.*

We claim that $P(d) \geq \frac{1}{d+2}$. We can prove this by induction. The base case is just $P(0) = \frac{1}{2}$. By induction, we can assume the claim holds up to $d-1$, i.e. $P(d-1) \geq 1/(d+1)$. Using the exercise, we have

$$P(d) \geq P(d-1) - \frac{1}{2}P(d-1)^2 \geq \frac{1}{d+1} - \frac{1}{2(d+1)^2} \geq \frac{1}{d+1} - \frac{1}{(d+1)(d+2)} = \frac{1}{d+2},$$

where we used that $2(d+1) \geq d+2$ for all $d \geq 0$.

Finally, the probability that the algorithm finds a minimum cut is at least $P(h)$, where $h = O(\log n)$ is the height of the recursion tree. By the claim we just proved, this probability is $P(h) \geq \frac{1}{h+2} \geq \frac{1}{O(\log n)}$, as we needed to show. □

To get an algorithm with constant probability of success, we just need to repeat the entire algorithm $O(\log n)$ times, and we get an algorithm with running time $O(n^2(\log n)^2)$.

It's not hard to see that the total number of leaves in the recursion tree is $\Theta(n^2)$. Another way to see this algorithm then is as a different way to repeat the contraction algorithm $\Theta(n^2)$ times to amplify the probability of success. Now the runs are no longer independent: different runs reuse some of the same contraction steps. This saves a lot in running time, and hurts us only a little bit in probability of success (which dropped from constant to $\frac{1}{O(\log n)}$.