CSE 521: Design and Analysis of Algorithms I **Fall 2020**

Lecture 1: Contraction Algorithm

Lecturer: Shayan Oveis Gharan *09-30-2020* *Scribe:*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

1.1 Introduction to Randomized Algorithms

Deterministic algorithms take input and produce output. In Randomized Algorithms, in addition to input algorithms take a source of random bits and makes random choices during execution - which leads behavior to vary even on a fixed input. For many problems a randomized algorithm is the simplest or fastest or both.

Let $p > 0$ be the probability that a (randomized) algorithm generates the correct (optimal) solution. It turns out that, even if p is much smaller than 1 we can obtain an algorithm with that succeeds with high probability just by executing the original algorithm independently many times. In particular, if we run the original algorithm k times, the probability that at least one copy succeeds is at least $1 - (1-p)^k$. For small values of p one can always approximate $1 - p$ with e^{-p}, and during this course we always use such an approximation. It follows that for $k = 100/p$, at least one copy finds the correct (optimal) solution with probability at least

$$1 - e^{-pk} = 1 - e^{-100}.$$

In other words, even if we have an algorithm with a small probability of success we can boost the success probability to a number very close to 1.

In this lecture, we describe a randomized algorithm for the minimum cut problem. Let us start with the definition of minimum cut problem. Let $G = (V, E)$ be a graph with $n = |V|$ vertices. Let

$$E(S, \overline{S}) := \{(u, v) : u \in S, v \notin S\},$$

be the set of edges that have exactly one endpoint in S. In the minimum cut problem, we want to partition the into two subsets which are joined together with *minimum* number of edges, i.e.,

$$\min_{\emptyset \subset S \subset V} |E(S, \overline{S})|.$$

1.2 Karger's Algorithm

In this lecture we will discuss Karger's [**Kar93**] and Karger-Stein's [**KS93**] algorithm for the minimum cut problem. We will show that the former finds the minimum cut in time $O(n^4)$ and the latter finds it in time $O(n^2)$ with high probability.

Before describing these algorithms, let us define a contraction procedure. Contraction of an edge (u, v) in G, merges the endpoints u and v to create a new (super) node uv. This reduces the total number of nodes in the graph by 1. All other edges which were previously attached to u or v are attached to the new (super) node uv. Note that this might lead to multiple parallel edges; in particular, if a node z has a edges to u and b edges to v, after contraction it will have $a + b$ edges to uv.

When we contract u, v we remove all edges connecting u to v from the graph; in other words, we do not include the loops.

Let k be the size of the minimum cut of G; fix a minimum cut $(S^*, \overline{S^*})$. In this section we design an algorithm that finds $(S^*, \overline{S^*})$ with probability at least $1/\binom{n}{2}$.

Let us start by describing the main idea. Suppose we choose a uniformly random edge e in G. What is the probability that $e \in E(S^*, \overline{S^*})$?

Claim 1.1. *The probability that a uniformly random edge is in $E(S^*, \overline{S^*})$ is at most $2/n$.*

Proof. Let e be a uniformly random edge in G. Obviously,

$$\mathbb{P}\left[e \in E(S^*, \overline{S^*})\right] = \frac{|E(S^*, \overline{S^*})|}{|E|} = \frac{k}{|E|}.$$

To give an upper bound we need to lower bound $|E|$. Here we use the hand-shake lemma. Let $d(v)$ be the degree of a vertex v. The hand-shake lemma says that in any graph G,

$$\sum_{v \in V} d(v) = 2|E|.$$

This is because in the LHS we count each edge twice. Now, we can lower bound $d(v)$ for any vertex v by k. This is because, for any v,

$$d(v) = |E(\{v\}, \overline{\{v\}})| \geq k.$$

In other words, the size of the cut separating v from the rest of the graph is at least k. It follows from the above two equations that $|E| \geq nk/2$. So,

$$\mathbb{P}\left[e \in E(S^*, \overline{S^*})\right] = \frac{k}{|E|} \geq \frac{k}{nk/2} = \frac{2}{n}.$$

□

Using the above lemma, if we choose $n/4$ edges of G uniformly at random, with probability $1/2$ none of them is in the min-cut $(S^*, \overline{S^*})$. Now, we can contract these $n/4$ edges. The new graph will have at most $3n/4$ vertices and we can recurse. After $O(\log n)$ steps, we get to a graph with just two super nodes. With probability at least $(1/2)^{O(\log n)}$ we do not contract any of the edges of $(S^*, \overline{S^*})$ throughout the process. So, the size of the cut separating the final two super-nodes is exactly k. In fact, one of these two super-nodes corresponds to S^* being contracted and the other one corresponds to $\overline{S^*}$.

In above we described the gist of the idea of Karger's algorithm. There are several missing points in the above description. First of all, when we recursively call the algorithm, we are recursively using the above claim. So, we need to make sure the size of the min-cut of G does not decrease when we contract an edge.

Fact 1.2. *For any graph G, when we contract an edge (u, v) the size of the minimum cut does not decrease.*

We leave the proof of the above fact as an exercise.

Secondly, when we choose $n/4$ edges uniformly at random many of them may be parallel, so after contraction the number of vertices of G will go down only by 1. To avoid running into these issues it suffices to contract edges one by one. That is, each time we choose a uniformly random edge of G and we contract it.

Theorem 1.3. *For any graph $G = (V, E)$ with n nodes and any min-cut $(S^*, \overline{S^*})$, Algorithm 1 returns $(S^*, \overline{S^*})$ with probability at least $\frac{2}{n(n-1)}$.*

Algorithm 1 Karger's Algorithm

for $i = 1 \to n-1$ **do**

Choose a uniformly random edge (u, v) and contract it, i.e., remove u, v, add a new node uv, connect all edges that go to u or v to the new node uv, also remove all loops.

end for

Return the number edges between the final two super-nodes as the size of the min-cut.

Proof. Let, A_i be the event that the edge picked in step i of the loop is not in $E(S^*, \overline{S^*})$. Observe that the algorithm succeeds in finding $(S^*, \overline{S^*})$ if $A_1, A_2, \ldots, A_{n-2}$ occur, i.e., if we never contract an edge of $E(S^*, \overline{S^*})$. So, we just need to lower bound $\mathbb{P}[A_1, A_2, \ldots, A_{n-2}]$. By Bayes rule we have,

$$\mathbb{P}[A_1, \ldots, A_{n-2}] = \mathbb{P}[A_1]\,\mathbb{P}[A_2|A_1]\,\mathbb{P}[A_3|A_1, A_2]\ldots\mathbb{P}[A_{n-2}|A_1, A_2, \ldots, A_{n-3}].$$

Now, by Claim 1.1 and Fact 1.2, for all i,

$$\mathbb{P}\left[\overline{A_{i+1}}|A_1, \ldots, A_i\right] \leq \frac{2}{n-i}.$$

This is because, at the $i+1$ step, there are only $n-i$ vertices in the graph. By Fact 1.2, the size of the min cut is at least k. So, the degree of each of these $n-i$ vertices is at least k, and the graph has at least $(n-i)k/2$ edges. Now, the above inequality follows by a similar reasoning as in Claim 1.1.

Therefore,

$$\begin{aligned}
\mathbb{P}[A_1, \ldots, A_{n-2}] &= \mathbb{P}[A_1] \geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\cdots\left(1 - \frac{2}{3}\right) \\
&= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{1}{3} \\
&= \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}.
\end{aligned}$$

□

The following corollary is immediate from the above theorem.

Corollary 1.4. *Any graph has at most* $\binom{n}{2}$ *min-cuts.*

Proof. Suppose there is a graph G that has more than $\binom{n}{2}$ min-cuts. Then, for one of those cuts, say $(S, \overline{S})$ the probability that Algorithm 1, finds $(S, \overline{S})$ is less than $1/\binom{n}{2}$. But, this is a contradiction. □

In fact, the above bound is tight. A cycle of length n has exactly $\binom{n}{2}$ min-cuts, one corresponding to each possible way to delete two edges of the cycle.

Runtime and Boosting Probability of Success. It is not hard to see that one can run Algorithm 1 in time $O(n^2)$, but as we proved above, the success probability of each execution is just $O(1/n^2)$. Using the boosting idea, we can boost the probability of success to $1 - 1/n$ by running $O(n^2 \log n)$ independent copies of the above algorithm and returning the best cut that any of the copies find. In the next section we describe a much faster algorithm due to Karger and Stein.

1.3 Karger-Stein Algorithm

Recall that Karger's algorithm only fails if it contracts an edge of the min-cut. Also, note that the probability that we contract an edge of the min-cut at the beginning is only $2/n$ while towards the end of the algorithm this probability goes up to a constant. In particular, in the very last step there is a probability of $1/3$ that we contract an edge of the min-cut.

The idea of Karger-Stein algorithm is to run multiple independent copies of the Karger's algorithm when the size of the Graph gets smaller. This idea kind of resembles the idea fault tolerant systems where one stores multiple copies of the data to decrease the probability of failure.

Let us describe Karger-Stein's algorithm.

Algorithm 2 Min-cut$(G = (V, E))$

Let $n = |V|$. If $n = 2$ return the unique cut separating the two nodes of G.
for $i = 1 \to n - n/\sqrt{2}$ **do**
  Choose a uniformly random edge and contract it.
end for
Let G' be the contracted graph. Call Min-cut(G') twice and return the best cut that any of these two copies find.

Let us divide the work of the algorithm into $O(\log n)$ phases; in the first phase the algorithm goes from n to $n/\sqrt{2}$, in the second phase it goes from $n/\sqrt{2}$ to $n/\sqrt{2}^2$ and so on. The number of copies are chosen such that the algorithm spends exactly the same amount of work $O(n^2)$ in each phase.

Let $T(n)$ be the time it takes to compute the min-cut of a graph of size n. Then,

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

We can use the master theorem to solve the above recurrence. But, usually it is easier to open it up a couple of times and see the pattern. We can write

$$\begin{aligned} T(n) &= O(n^2) + 2O((n/2^{1/2})^2) + 4O((n/2^1)^2) + \dots \\ &= O(n^2) + O(n^2) + O(n^2) + \dots = O(n^2 \log n). \end{aligned}$$

It remains the calculate the probability of success. In the next theorem we show that the algorithm succeeds with probability $\Omega(1/\log n)$. This shows that to boost the probability of success to $1 - 1/n$ it is enough to run $\log^2 n$ independent copies of the above algorithm. Therefore, the algorithm finds the min-cut with probability $1 - 1/n$ in time $O(n^2 \log^3 n)$.

Theorem 1.5. *For any min-cut $(S^*, \overline{S^*})$, Algorithm 2 finds this cut with probability at least $1/2 \log n$.*

Proof. Suppose we call Min-cut function on G with n vertices. By an analysis similar to Theorem 1.3, the probability that we do not contract any edge of $(S^*, \overline{S^*})$ in the $n - n/\sqrt{2}$ steps of the loop is at least

$$\frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdots \frac{n/\sqrt{2}-2}{n/\sqrt{2}} \approx \frac{(n/\sqrt{2})^2}{n^2} = 1/2.$$

The algorithm succeeds in finding the min-cut $(S^*, \overline{S^*})$ if Min-cut(G) does not contract an edge of $(S^*, \overline{S^*})$ in its for loop and at least one of the two copies succeeds in finding the cut. We prove inductively that for

any graph with n vertices the probability of success is at least $1/2 \log n$. Assuming by induction hypothesis, that the algorithm succeeds on G' with probability at least $p \geq \frac{1}{2\log(n/\sqrt{2})}$. Here the logs are all in base 2.

The probability that the algorithm succeeds in G is at least

$$\frac{1}{2}(1-(1-p)^2)) = \frac{1}{2}(2p-p^2) = p - p^2/2$$

Note that $(1-p)^2$ is the probability that both of the two independent copies fail in finding the cut. So, $1-(1-p)^2$ is the probability that at least one of them succeed. The $1/2$ ratio is the probability that in the first $n - n/\sqrt{2}$ iterations of the loop we succeed and we do not contract an edge of $S^*, \overline{S^*}$.

So, it is enough to show that

$$p - p^2/2 \geq \frac{1}{2\log n}.$$

In the worst case we have $p = \log(n/\sqrt{2})$. So, we need to show

$$\frac{1}{2\log(n/\sqrt{2})} - \frac{1}{8\log(n/\sqrt{2})^2} \geq \frac{1}{2\log n}.$$

Equivalently, it is enough to show

$$\frac{1}{\log(n/\sqrt{2})} - \frac{1}{\log n} \geq \frac{1}{4\log(n/\sqrt{2})^2}$$

The latter holds because,

$$\frac{1}{\log(n/\sqrt{2})} - \frac{1}{\log n} = \frac{\log n - \log(n/\sqrt{2})}{\log n \log(n/\sqrt{2})} = \frac{1/2}{\log n \log(n/\sqrt{2})} \geq \frac{1}{4\log(n/\sqrt{2})^2}.$$

□

References

[Kar93] D. R. Karger., "Global min-cuts in RNC, and other ramifications of a simple min-out algorithm," in *SODA.* , 1993, pp. 21–30.

[KS93] D. R. Karger and C. Stein, "An $O(n^2)$ algorithm for minimum cuts", in *STOC.*, (1993)