CS271 Randomness & Computation	Spring 2020

Lecture 11: February 25

Instructor: Alistair Sinclair

Disclaimer: *These notes have not been subjected to the usual scrutiny accorded to formal publications. They may be distributed outside this class only with the permission of the Instructor.*

11.1 Network Reliability

Consider a connected undirected graph G with n vertices and m edges, where each edge has some probability p of failing. What is the probability that G becomes disconnected under random, independent edge failures? This can also be viewed as a counting problem, except that each item now has an associated weight: in particular, we want to compute the sum of the weights of all disconnected subgraphs of G, where the weight of a subgraph with t fewer edges than G is $p^t(1-p)^{m-t}$. We can also generalize the problem to allow a different failure probability for each edge.

> **Input:** A connected graph $G = (V, E)$, and edge failure probabilities p_e for each edge $e \in E$.
>
> **Goal:** Compute $p_{\text{fail}} = \Pr[\text{G becomes disconnected when each edge } e \text{ fails independently with probability } p_e]$.

This problem is $\#\mathcal{P}$-hard, even in the special case $p_e = p = 1/2 \ \forall e \in E$.

Definition 11.1 *A fully polynomial randomized approximation scheme (FPRAS) on input (G,$\{p_e\}$,ϵ) outputs a value Z such that* $\Pr[(1-\epsilon)p_{\text{fail}} \leq Z \leq (1+\epsilon)p_{\text{fail}}] \geq \frac{3}{4}$*, and runs in time polynomial in* $(n, \frac{1}{\epsilon})$.

Theorem 11.2 (Karger [Kar95]) *There exists a FPRAS for network reliability (for any set of edge-dependent failure probabilities $\{p_e\}_{e \in E}$).*

We begin with a high-level sketch of the algorithm. For simplicity, we will restrict attention to the case that all edge probabilities are equal, i.e., $p_e = p \ \forall e \in E$. Let c be the size of a minimum cut in G. Then clearly we have $p_{\text{fail}} \geq p^c$, since if all of the edges of any cut fail, G becomes disconnected. Then, we can make the following observations:

1. If $p^c \geq \frac{1}{n^4}$, then a naive Monte Carlo approach works. Namely, suppose we simply pick a random subgraph of G by removing each edge independently with probability p, and set $X_i = 1$ if the subgraph is disconnected, and $X_i = 0$ otherwise. Then X_i is an unbiased estimator of $\mu = p_{\text{fail}}$, and by the Unbiased Estimator theorem, we only need $O(\frac{1}{\mu\epsilon^2}) = O(n^4\epsilon^{-2})$ trials to achieve the desired error bound.
2. Otherwise, if $p^c < \frac{1}{n^4}$, then, for $\alpha = 2 + \frac{1}{2}\log_n(2/\epsilon)$, we will prove that
$$\Pr[\text{some cut of size} \geq \alpha c \text{ fails}] \leq \epsilon p^c \leq \epsilon p_{\text{fail}}.$$
Therefore, we can effectively ignore cuts of size $\geq \alpha c$, by absorbing the resulting error into ϵ. (More precisely, we will get an estimate within $(1 \pm \epsilon)^2$, which is no worse than $(1 \pm 3\epsilon)$; so we just replace ϵ by 3ϵ.)

3. We say a cut is an α-**minimum cut** if it has size $\leq \alpha c$. Then we have the following claim and corollary:

   Claim 11.3 *There are at most $n^{2\alpha} = \frac{2n^4}{\epsilon}$ α-minimum cuts, and these cuts can be enumerated in time polynomial in $(n, \frac{1}{\epsilon})$.*

   Corollary 11.4 *The probability that an α-minimum cut fails can be expressed as the solution to a probabilistic DNF problem as follows:*

   $$\Pr[\textit{some } \alpha\textit{-minimum cut fails}] = \Pr\Big[\bigvee_{i=1}^{t} (x_{e_{i_1}} \wedge x_{e_{i_2}} \wedge \ldots \wedge x_{e_{i_r}})\Big],$$

   where the OR is over all $t \leq 2n^4/\epsilon$ α-minimum cuts, $\{e_{i_1}, \ldots, e_{i_r}\}$ are the edges of the i^{th} cut and

   $$x_{e_j} = \begin{cases} T & \text{with probability } p; \\ F & \text{otherwise.} \end{cases}$$

   Moreover, this formula can be constructed in time polynomial in $(n, \frac{1}{\epsilon})$.

   With the above corollary, we can apply the Karp/Luby algorithm for probabilistic DNF from the previous lecture to get a FPRAS for computing the probability that an α-minimum cut fails. In light of items 1 and 2, this gives us an FPRAS for Network Reliability.

It remains to prove the key Claim 11.3, and also the statement made in item 2 above (which in fact also follows from Claim 11.3). We prove the claim in the next subsection.

11.2 Proof of Claim 11.3

We first describe a randomized algorithm (also due to Karger [Kar93]) for finding a minimum cut in a connected graph $G = (V, E)$. (Note that this algorithm can be used to find the value of c that is required in step 1 of the above algorithm.) We'll call this algorithm *RMinCut*. From it, we will easily obtain an upper bound on the number of minimum cuts in any graph.

while $|V| > 2$ **do**
  Choose an edge $\{u, v\} \in E$ uniformly at random.
  Merge u and v, maintaining all edges from either of them to other vertices.
Return the remaining cut.

Note that the algorithm actually maintains a multigraph, since during a merge operation multiple edges may be created. When a random edge is picked, we view multiple edges as distinct.

Theorem 11.5 *Let $C \subset E$ be any minimum cut. Then* $\Pr[\textit{RMinCut returns } C] \geq \binom{n}{2}^{-1}$.

Proof: Say that C is *hit* at stage i if one of its edges $\{u, v\}$ is selected and collapsed at stage i in $RMinCut$. Observe that no vertex of G can have fewer than c neighbors; otherwise the cut disconnecting just that vertex would have size less than c. Hence, $|E(G)| \geq \frac{nc}{2}$ and

$$\Pr[C \text{ is hit in round } 1] \leq \frac{c}{nc/2} = \frac{2}{n}.$$

Similarly,

$$\Pr[C \text{ is hit in round } i+1 \mid C \text{ survives rounds } 1,...,i] \le \frac{c}{(n-i)c/2} = \frac{2}{n-i}.$$

Therefore,

$$\begin{aligned}\Pr[C \text{ survives all rounds}] &\ge (1-\frac{2}{n}) \times (1-\frac{2}{n-1}) \times ... \times (1-\frac{2}{3}) \\ &= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times ... \times \frac{1}{3} \\ &= \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}.\end{aligned}$$

■

Note that $O(n^2)$ trials of *RMinCut* are required to be confident that we have found a minimum cut. An obvious implementation takes time $O(n^2)$ per trial, for an overall running time of $O(n^4)$.
Exercise: *Devise a cleverer, recursive implementation of RMinCut which achieves $O(n^2 \log n)$ overall running time.*
One may compare this with the approach of finding an s-t minimum cut for any fixed s and all t using a network flow algorithm, which naively would take overall time about $O(n^4 \log n)$ (for $O(n)$ flow computations each of cost $O(n^3 \log n)$).

For our purposes, a more important consequence of the above is the following

Corollary 11.6 *For any graph G, the number of minimum cuts is at most $\binom{n}{2}$.*

This follows immediately from the fact that each distinct minimum cut is output with probability at least $\binom{n}{2}^{-1}$, and these are disjoint events.

Now let us focus on α-minimum cuts.

Corollary 11.7 *The number of α-minimum cuts is at most $n^{2\alpha}$ in any graph G.*

Proof: We know that by definition, an α-minimum cut C has size at most αc. As in the proof of Theorem 11.5, we have

$$\Pr[C \text{ is hit in round } 1] \le \frac{\alpha c}{nc/2} = \frac{2\alpha}{n}$$

and

$$\Pr[C \text{ is hit in round } i+1 \mid C \text{ survives rounds } 1,...,i] \le \frac{\alpha c}{(n-i)c/2} = \frac{2\alpha}{n-i},$$

so that

$$\Pr[C \text{ survives until } 2\alpha \text{ vertices remain}] \ge \binom{n}{2\alpha}^{-1}.$$

(It is necessary to employ Γ functions to make sense of $\binom{n}{2\alpha}$ if 2α is not an integer, but we won't dwell on this detail here.)

Now define a process K as follows:

- apply the *RMinCut* routine until 2α vertices remain
- pick a random cut in the remaining multigraph

Then we have

$$\begin{aligned}
\Pr[C \text{ survives } K] &= \Pr[C \text{ survives until } 2\alpha \text{ vertices remain}] \times \Pr[C \text{ survives random cut}] \\
&\geq \frac{1}{\binom{n}{2\alpha}} \times \frac{1}{2^{2\alpha-1}} \\
&\geq \frac{(2\alpha)!}{2^{2\alpha}} \cdot \frac{1}{n^{2\alpha}} \geq \frac{1}{n^{2\alpha}},
\end{aligned}$$

which proves the corollary. ■

This is the first part of the Claim 11.3 of the algorithm. It remains only to enumerate the α-minimum cuts of G. This is achieved via the coupon-collector paradigm: if there are n bins and balls are thrown independently and uniformly, how many balls must be thrown to have every bin contain at least one ball? We have

$$\Pr[\text{at least } (n \log n + an) \text{ balls must be thrown}] \approx e^{-e^{a}}$$

for any fixed a, so with very high probability $O(n \log n)$ balls suffice. Thus, if we repeatedly run process K above, after $O(n^{2\alpha} \log(n^{2\alpha})) = O(n^4(\log n + \log \epsilon^{-1})/\epsilon)$ attempts we will have enumerated all α-minimum cuts with high probability. (The small probability of failure can be absorbed into the error probability of our FPRAS.)

11.3 Proof of item 2

To complete the analysis of Karger's algorithm, we need to prove item 2 in the high-level sketch given earlier. For convenience, let $\delta > 2$ be such that $p^c = n^{-(2+\delta)}$. Let $C_1, C_2, \ldots$ be an enumeration of the cuts of size at least αc, and for each i, let $c_i = |C_i|$. We will assume that $c_1 \leq c_2 \leq c_3 \leq \ldots$. Let us divide the analysis into two parts.

(i) We'll consider the first $n^{2\alpha}$ cuts, then the remainder in sequence. We can also assume, without loss of generality, that there are at least $n^{2\alpha}$ cuts; otherwise, the argument below gives an even better bound.

Note that for each $i \leq n^{2\alpha}$, $\Pr[\text{all edges in } C_i \text{ fail}] \leq p^{\alpha c}$. Hence,

$$\Pr[\bigvee_{i=1}^{n^{2\alpha}} C_i \text{ fails}] \leq n^{2\alpha} p^{\alpha c} = n^{2\alpha} n^{-(2+\delta)\alpha} = n^{-\delta\alpha}.$$

This takes care of the initial sequence of cuts.

(ii) For any $\beta > 0$, we know that there are no more than $n^{2\beta}$ cuts of size at most βc, by Corollary 11.7; that is, $c_{n^{2\beta}} \geq \beta c$. Writing $k = n^{2\beta}$, this translates to $c_k \geq \frac{c}{2} \log_n k$, so

$$p^{c_k} \leq p^{\frac{c}{2} \log_n k} = n^{-(2+\delta)(\log_n k)/2} = k^{-(1+\delta/2)}.$$

Thus,

$$\Pr[\bigvee_{i > n^{2\alpha}} C_i \text{ fails}] \leq \sum_{k > n^{2\alpha}} k^{-(1+\delta/2)} \leq \int_{n^{2\alpha}}^{\infty} x^{-(1+\delta/2)} dx = \frac{2}{\delta} n^{-\delta\alpha} < n^{-\delta\alpha}.$$

Putting (i) and (ii) together, we get

$$\Pr[\text{some cut of size} \geq \alpha c \text{ fails}] \leq 2n^{-\delta\alpha}.$$

Now with our choice of $\alpha = 2 + \frac{1}{2}\log_n(2/\epsilon) \geq 1 + \frac{2}{\delta} + \frac{1}{\delta}\log_n(2/\epsilon)$, this yields

$$\Pr[\text{some cut of size} \geq \alpha c \text{ fails}] \leq \epsilon n^{-(2+\delta)} = \epsilon p^c.$$

This concludes the argument for item 2 and the analysis of the algorithm claimed in Theorem 11.2.

Remarks:

1. A different, theoretically more efficient algorithm for the same problem, incorporating some of the same ideas but dispensing with the expensive step of cut enumeration, was recently obtained by Karger [Kar16].

2. This algorithm can be derandomized in every aspect except Part 1, the naive Monte Carlo routine. It is possible to derandomize both the enumeration of cuts and the Karp-Luby algorithm in polynomial time, but it is not known how or if the simple Monte Carlo method can be derandomized efficiently.

3. Note that this FPRAS does not provide a good estimate of $p_{\text{success}} = (1 - p_{\text{fail}})$, which is interesting when the probability p of edge failure is large (a less important scenario in the Network Reliability context, but mathematically interesting). Very recently, Guo and Jerrum have provided a FPRAS for p_{success}, using quite different methods [GJ17]. We may discuss this algorithm later in the course.

References

[GJ17] H. Guo and M. Jerrum, “A polynomial-time approximation algorithm for all-terminal network reliability,” preprint, arXiv:1709.08561 [cs.DS], 2017.

[Kar93] D.R. Karger, “Global Min-Cuts in $\mathcal{RNC}$ and other ramifications of a simple Min-Cut algorithm,” *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Jan. 1993, pp. 21–30.

[Kar95] D.R. Karger, “A randomized fully polynomial approximation scheme for the all terminal network reliability problem”, *In Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC)*, 1995, pp. 11-17

[Kar16] D.R. Karger, “A fast and simpler unbiased estimator for network (un)reliability,” *Proceedings of the 48th annual Symposium on the Foundations of Computer Science (FOCS)*, 2016, pp. 635–644.